University of Alberta

ARMarkov Model-based Predictive Control: Design and Stable Tuning

by

Mst Kamrunnahar

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

in

Process Control

Department of Chemical and Materials Engineering

Edmonton, Alberta
Fall 2001

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-68954-9

Canada

University of Alberta

Library Release Form

Name of Author: Mst Kamrunnahar

Title of Thesis: ARMarkov Model-based Predictive Control: Design and Stable Tuning

Degree: Doctor of Philosophy

Year this Degree Granted: 2001

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Mst Kamrunnahar
CME 536
University of Alberta
Edmonton, AB
Canada, T6G 2G6

Date: July 31, 2001

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **ARMarkov Model-based Predictive Control: Design and Stable Tuning** submitted by Mst Kamrunnahar in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in *Process Control.*

. . . . . .

Dr. D. Grant Fisher (Supervisor)

. . . . . .

Dr. Biao Huang (Co-supervisor)

. . . . . .

Dr. Max Meng

. . . . . .

Dr. J. Fraser Forbes

. . . . . .

Dr. Sachin Patwardhan (External Examiner)

Date: July 30, 2001

To my parents

Abstract

A design and tuning procedure for Model-based Predictive Controllers (MPC) was developed based on a **ARMarkov model** structure in which the user specifies the number of Markov (impulse response) parameters required to define the process time-delay and the initial, fast dynamics of the time domain response. A low order, parametric ARX model is then specified to model the slower dynamics of the remaining process dynamics. The advantages of this dual model structure are that it does not require accurate, a priori specification of the process time-delay or model order and includes provision for disturbance modelling. Furthermore, the statistical properties of the identified Markov parameters are better than the same properties of parameters generated using other linear regression ID methods. The design of the **controller** follows traditional MPC techniques but the use of the ARMarkov "dual model" and a separate disturbance model results in a flexible formulation that includes DMC and GPC as special cases and permits independent tuning of the closed-loop servo and regulatory responses.

An **on-line tuning** methodology is developed that linearly combines the outputs of any two individual controllers using a continuous parameter α. As α is varied from 0 to 1 the state feedback gain is a convex combination of the gains of the two individual, limiting controllers and the system properties vary continuously between the limits defined by each controller used individually, e.g. from conservative, robust control to aggressive, high-performance control. If the conditions stated in Lemmas 11.1 and 11.2 are met, the closed-loop system is stable using the combined output for all $0 \leq \alpha \leq 1$.

When the two MPC's differ only in the specification of the control horizon ($M = m$ and $M = m + n$) or the steady-state output weighting, the second controller output can be obtained by a simple extension of the first controller calculation and hence there is no significant increase in computational requirements.

The performance monitoring approach simultaneously provides closed-loop time domain performance indices (*e.g.* rise/settling time) plus the interactor matrix and sensitivity /complementary-sensitivity functions. State space based performance benchmarks such

as linear quadratic gaussian (LQG) are also formulated.

Simulated and experimental applications are included to illustrate the theoretical developments in this thesis.

Acknowledgements

I would like to express my gratitude and thanks to Prof. D.G. Fisher for his kind supervision, technical advice and guidance throughout my PhD research. I thank him for his constant support and encouragement during my personal problems.

I am thankful to my thesis co-supervisor Prof B. Huang for his technical help, personal encouragement and advice.

Many thanks to Professors Shah, Wood, Forbes and Chen for teaching me the control theory that enabled me to understand and pursue my research with enthusiasm.

I am happy that I got the opportunity to work in an innovative and friendly environment like the Computer Process Control group. Many thanks to the group members such as Laksh, Anand, Arun, Misha, Donguang, Bhushan, Lanny, Ramesh, Ashish, Aseema, Rohit, Haitao, Raghu, Harigopal and many others. Thanks to Arun for being such a special friend and an excellent study partner especially during our PhD qualifying exam. I will remember Bhushan for his witty comments which added humor to our many technical discussions.

I am grateful to my husband Aman for being such a wonderful person giving me continuous support during my PhD research. He deserves special thanks for encouraging me and cheering me up during the frustrating and unproductive times.

I also wish to thank my parents for giving me love and care and leading me towards higher education.

Thanks to Tusher, Samina, Nazmul, Mahbub bhai, Tariq, Shoukat, Shawkat and Seena for being such special friends and helping me personally when I needed it the most. I will always remember the good times we had.

I am thankful to the Chemical and Materials Engineering Department for giving me the opportunity to study here, to the departmental office secretaries and the computing staff for their help. Thanks particularly to Bob Barton for his help with the computers.

I would like to thank the U of Alberta-BUET linkage project with financial support from CIDA for providing me with the scholarship to pursue my PhD studies. Thanks to the people working in the project for being very helpful.

Contents

1 Introduction 1

1.1 Overview of Model-based Predictive Control (MPC) 2

1.1.1 Basic Principles and Advantages 2

1.1.2 Historical Background and Present/Future Research Directions 4

1.2 Research Objectives 5

1.2.1 Motivation 5

1.2.2 General Objectives 8

1.3 Contents and Contributions of the Thesis 9

I ARMarkov Identification 12

2 Estimation of Markov Parameters using ARMarkov Identification[1] 13

2.1 Introduction 13

2.2 Linear Regression Models and Least-squares Methods 16

2.2.1 ARX Model 17

2.2.2 Finite Impulse Response (FIR) Model 18

2.3 The ARMarkov Method 19

2.3.1 ARMarkov/LS Identification Algorithm 21

2.4 Extension of the ARMarkov Method to MIMO Systems 22

2.5 Recursive ARMarkov/LS Identification 23

2.5.1 SISO Systems 24

2.5.2 MIMO Systems 24

2.6 Simulation Examples 25

[1]This chapter is a part of the paper published in the *Chemical Engineering Science*, May 2000. It was also presented at the 1998 CSCHE Conference, London, Ontario, Canada.

2.6.1 Open-Loop SISO Systems 25
2.6.2 Open-Loop MIMO Systems 25
2.7 Model Identification of a Pilot Scale CSTH Using Experimental Data 27
2.8 Conclusions 29

3 Estimation of Parametric Residual Model Using AUDI **31**
3.1 Introduction 31
3.2 Introduction to the AUDI Method 32
3.2.1 Regressor Vector and Information Matrix for the AUDI Method 33
3.2.2 Decomposition of IAM or DPMM and Parameter Estimation in the AUDI Algorithm 34
3.2.3 Simulation Example 36
3.3 Re-arrangement of the Regressor Vector for the ARMarkov Model . . 37
3.4 AUDI Formulation of the ARMarkov Model 38
3.4.1 Solution of the Parameter Vector in the AUDI-ARMarkov method 39
3.4.2 Loss Functions with Multiple Model Order 39
3.5 Simulation Example 40
3.6 Conclusions 41

4 Statistical Analysis of the ARMarkov Parameter Estimates[2] **42**
4.1 Introduction 42
4.2 Consistency of the Least-squares Estimates 43
4.2.1 Consistency Under Closed-loop Condition 47
4.3 Covariance of the Estimated Parameters 47
4.3.1 Covariance Estimates for Systems with White Noise 47
4.3.2 Covariance Estimates for Systems with Non-white Noise . . . 49
4.4 Confidence Intervals on the Parameter Estimates 50
4.5 Simulation Examples 52
4.5.1 Variance 52
4.5.2 Confidence bounds 53
4.6 Conclusions 54

[2]Contents of this chapter was presented at the 49th CSCHE conference, Saskatoon, Canada, Oct. 1999 and is a part of the paper published in the Chemical Engineering Science in May 2000.

5 Interactor Matrix Estimation from Markov Parameters[3] 55

5.1 Introduction . . . 55

5.2 The Interactor Matrix . . . 56

5.3 Estimation of the Unitary Interactor Matrix . . . 58

5.3.1 Determination of the Order of the Interactor Matrix . . . 58

5.3.2 Estimation of the Interactor Matrix . . . 59

5.4 Markov Parameters from the ARMarkov Method for Interactor Estimation . . . 59

5.5 Interactor Matrix for Closed-Loop Systems . . . 60

5.6 Simulation Examples . . . 62

5.6.1 Open-Loop MIMO Systems . . . 62

5.6.2 Closed-loop Systems . . . 64

5.7 Conclusions . . . 65

II Model Predictive Controller Design 67

6 Design and Formulation of an ARMarkov Model-based Predictive Controller (ARM-MPC)[4] 68

6.1 Introduction . . . 68

6.2 The Extended ARMarkov Model . . . 70

6.3 Predictive Control Design Using the Extended ARMarkov Model . . . 72

6.4 Special Cases of ARM-MPC . . . 75

6.4.1 GPC structure: . . . 76

6.4.2 DMC structure: . . . 76

6.5 Simulation Examples Using an ARM-MPC . . . 77

6.5.1 Comparison of Linear Systems . . . 79

6.6 Experimental Evaluation of ARM-MPC on a Pilot Scale Process . . . 80

6.7 Conclusions . . . 82

[3]Contents of this chapter was presented at the 1998 CSCHE conference in London, Ontario, Canada and was also published in the *Chemical Engineering Science*, May 2000.

[4]This chapter was presented at the ADCHEM 2000 conference, June 2000 and is a part of the paper to be published in the *Journal of Process Control*.

7 Disturbance Prediction in the ARM-MPC[5] 83

7.1 Introduction 83

7.2 Extended ARMarkov Model With Observer Polynomial 85

7.3 ARM-MPC With Observer Polynomial 86

7.3.1 Flexibility of Disturbance Predictions 89

7.3.2 Limiting Case of ARM-MPC 90

7.4 Simulation Examples 90

7.5 Conclusions 92

8 Dual-Model State-space ARM-MPC 95

8.1 Introduction 95

8.2 An Overview of the State-space Formulation of MPC 97

8.2.1 Model Formulation 97

8.2.2 State Estimation 98

8.2.3 Output Prediction 99

8.2.4 Control Move Calculation 99

8.2.5 Closed-loop Formulation 100

8.2.6 Closed-loop Properties 101

8.3 State Space ARM-MPC based on the Input-output ARMarkov Model 101

8.3.1 Feedback Observer Design 104

8.3.2 Closed-loop Formulation 105

8.4 Recursive Control Move Calculation for Increasing Control Horizon . 106

8.5 Steady-state Weighting, γ_∞ 109

8.5.1 Steady-state Output Prediction 110

8.5.2 Calculation of steady-state Gain 110

8.5.3 Control Calculations Using Steady-state Error Weighting . . . 113

8.5.4 Simultaneous Steady-state and Control Weighting 114

8.5.5 Feedback Gain Update Using $\gamma_{\infty 2}$ 117

8.6 Examples 119

8.7 Conclusions 120

[5]Material in this chapter forms a part of the paper published in the Journal of Process Control.

9 MPC Design Using a Classical State-Space Model Generated from Markov Parameters 123

9.1 Introduction 123

9.2 State-Space Model From the Markov Parameters 124

9.2.1 Eigensystem Realization Algorithm (ERA) 125

9.2.2 Selection of Model Order When Using Noise Corrupted Data . 127

9.3 State-Space MPC Design 128

9.3.1 Augmented State-Space Model 128

9.3.2 State Estimation 129

9.3.3 Output Prediction and Control Move Calculation 129

9.3.4 Feedback Observer or State Estimator Design 131

9.3.5 Closed-loop Formulation of the State-space MPC 132

9.4 Simulation Results 133

9.5 Experimental Evaluation of Classical MPC on a Pilot Scale Process . 133

9.6 Conclusions 135

III Analysis of ARM-MPC 137

10 Performance and Robustness Analysis of ARM-MPC 138

10.1 Introduction 138

10.1.1 Performance Assessment 138

10.1.2 Robustness Analysis 139

10.1.3 Ultimate Objective 140

10.2 Performance Assessment Using a Minimum Variance Benchmark . . . 141

10.2.1 Filtering and CORrelation (FCOR) Algorithm 142

10.3 User-specified Benchmarks for Closed-loop Performance Analysis . . . 142

10.4 Performance Analysis Using Closed-loop ARMarkov Identification . . 143

10.4.1 The ARMarkov Identification 143

10.5 Performance Analysis Using Linear Quadratic Gaussian (LQG) Control as the Benchmark 150

10.6 Robustness Analysis of ARM-MPC 154

10.6.1 Closed-loop ARMarkov model and the Small Gain Theorem (SGT) . . . 157
10.6.2 Matrix Perturbation Method . . . 159
10.7 Conclusions . . . 166

11 Stable On-line Tuning of ARM-MPC for Performance-Robustness Trade-offs **168**
11.1 Introduction . . . 168
11.2 Continuous Control Action . . . 169
11.3 Stability of the $\alpha-$controller . . . 170
11.3.1 Controller Gain Calculation . . . 171
11.3.2 Stability of the $\alpha-$MPC . . . 171
11.4 Stable On-line tuning Using the α Parameter . . . 173
11.5 Controller Design Using a Continuous Control Horizon . . . 174
11.5.1 Recursive Calculation of Control Outputs for Two Control Horizons . . . 175
11.5.2 Continuous Control Horizon from $M = m$ to $M = m + n$ . . . 176
11.5.3 Simulation Results . . . 178
11.6 On-line Tuning Using Steady State Weighting . . . 179
11.6.1 Examples . . . 182
11.7 Trade-off Using a Combination of Steady-state Error Weighting and Control Weighting . . . 183
11.7.1 Feedback Controller Gain Update Using $\gamma_{\infty 2}$ . . . 185
11.7.2 Equivalence with the $\alpha-$tuning . . . 186
11.8 On-line Tuning Surface . . . 186
11.8.1 Simulation Examples . . . 188
11.9 Conclusions . . . 189

12 Conclusions and Future Work **193**
12.1 Contributions . . . 193
12.2 Overall Conclusion and Practical Significance . . . 198
12.3 Recommendations for Future Work . . . 199

Bibliography 201

A Factorization of Matrices in the AUDI Formulation 209

A.1 LDL^T and UDU^T factorization for the DPMM and IAM 209

List of Figures

1.1 Classical MPC Structure 7
1.2 MPC System Supervisor 8
1.3 Dual-model Representation of a Process Step Response 9

2.1 Impulse Response of a Third Order System 26
2.2 Impulse Response of a 2x2 MIMO System. 27
2.3 Schematic diagram of a pilot scale CSTH. 29
2.4 Experimental input-output data from a pilot scale CSTH. 30
2.5 ARMarkov model validation for a pilot scale CSTH. 30

3.1 Loss functions for different model orders in the AUDI formulation. . 36
3.2 Loss functions for different parametric model orders in the AUDI formulation of the ARMarkov model. 40
3.3 Actual and estimated Markov parameters using the AUDI formulation of the ARMarkov model. 41

4.1 Confidence Bounds on Markov Parameters 53

5.1 Closed-loop System. 61
5.2 Errors in the estimated coeffiecients of the interactor 64

6.1 Step Tracking and Disturbance Rejection using GPC ARM-MPC with a Type-I Disturbance Model. A unit step disturbance was added at time 100. 78
6.2 Comparison of Responses from DMC, GPC and ARM-MPC with a Type-1 disturbance model. In (a) 'solid'→ setpoints and 'dash'→ outputs. A unit step disturbance was added at time 100. 79

6.3 Responses from a pilot scale CSTH using ARM-MPC. 81

7.1 ARMarkov model-based predictive controller (ARM-MPC) using the separated disturbance predictor (SDP). 88

7.2 Comparison of responses using DMC, GPC and ARM-MPC with a Type-1 disturbance model in DMC and with an observer polynomial, $C = 1 - 0.8z^{-1}$ in GPC & ARM-MPC. In (a) 'solid'→ setpoints and 'dash'→ outputs. A unit step disturbance was added at time 100. . 91

7.3 Comparison of responses from DMC, GPC and ARM-MPC with a Type-1 disturbance model. In (a) 'solid'→ setpoints and 'dash'→ outputs. A unit step disturbance was added at time 100. 92

7.4 Responses from an ARMAX process using GPC and ARM-MPC with an observer polynomial, $C = 1 - 0.8z^{-1}$. In (a) 'solid'→ setpoints and 'dash'→ outputs. 94

8.1 Step Tracking and Disturbance Rejection using ARM-MPC with a Type-1 Disturbance Model. A unit step disturbance was added at time 100. 120

8.2 Process response by changing control horizon on-line. 121

8.3 Set-point tracking for simultaneous steady state error and control weighting using a single parameter, $\gamma_{\infty 2}$. 122

9.1 Closed-loop MPC system with disturbance predictor. 133

9.2 Step resposes for the actual process and the state space model. . . . 134

9.3 Performance of the state space MPC developed using the ARMarkov/ERA model. 135

9.4 Step tracking performance of state space MPC implemented on a pilot scale CSTH (2×2 MIMO system). 136

10.1 Closed-loop system 145

10.2 Markov parameters using CL ARMarkov method. 150

10.3 (a) Sensitivity and (b) complementary sensitivity functions using the CL ARMarkov method. 151

11.14 Step tracking performace of three different controllers. 191
11.15 Step tracking performance of the α-controller for different values of α. 192

List of Tables

1.1 Overview of Thesis Work on MPC 11

2.1 Markov Parameter blocks of a 2x2 MIMO system 28

3.1 Parametres corresponding to different models in the AUDI formulation 37
3.2 Parametres corresponding to different models in the AUDI formulation 37

4.1 Variance with white noise disturbances 52
4.2 Variance with non-white noise disturbances 53

5.1 Interactor Matrix for a 2x2 First order System 63
5.2 Coefficients in the Interactor Matrix 64
5.3 Interactor Matrix from Closed-loop Data 65

6.1 Close-loop Performance of Different Controllers with $C = 1$ 80

7.1 Input Variances and Performance Costs for Different Controllers with $C = 1 - 0.8z^{-1}$ 93
7.2 Input Variances and Performance Costs for Different Controllers with $C = 1 - 0.8z^{-1}$ 93

10.1 Comparison of the Advantages of Different Performance Assessment Methods 149
10.2 Factorization of the uncertainty matrix for different MPMs 164

11.1 Robustness bounds of ARM-MPC using different γ_∞ 182

Chapter 1

Introduction

Model-based Predictive Controller (MPC) has been widely accepted by academia and used in industrial applications because of its ability to handle constraints and large scale multivariable processes. Research on MPC design and analysis started in the 1970's and is continuing because of the ongoing demand for improved performance and robustness. The first step in model-based predictive controller design is to obtain a process model that is well suited for the controller to be designed. The controller can then be designed to optimize a user-specified performance index. However, when the designed controller is implemented on the actual process, there is always some performance loss due to model plant mismatch (MPM), known/unknown disturbances on the process, measurement errors, non-linearity and so on. Therefore, after implementing the controller there is always a need for controller tuning and sometimes even redesign. The need for controller tuning or re-design is usually established by a performance/robustness analysis of the controlled system. Therefore, *process modeling*, *controller design* and *analysis/tuning* are the major concerns in MPC applications. Each of these areas is addressed in this thesis.

An overview of MPC including past, present and future research directions is given in Section 1.1. The motivation for and the objectives of this thesis are described in Section 1.2 followed by an outline of the thesis organization and contributions in Section 1.3.

1.1 Overview of Model-based Predictive Control (MPC)

1.1.1 Basic Principles and Advantages

MPC has been defined by different researchers in different ways. Garcia *et al.* (1989)defined MPC as "that family of controllers in which there is a direct use of an explicit and separately identifiable model". Eaton and Rawlings (1992)said this definition was too general and identified the defining feature of MPC as "a repeated optimization of an open-loop performance objective over a finite horizon extending from the current time into the future". Several other formulations and different names for MPC can be found in the literature. Among them Dynamic Matrix Control (DMC) by Cutler and Ramaker (1980), Identification and Command (IDCOM) by Richalet *et al.* (1978), Model Algorithm Control (MAC) by Rouhani and Mehra (1982), Internal Model Control (IMC) by Garcia and Morari (1982), multivariable optimal constrained control algorithm (MOCCA) by Sripada and Fisher (1985), Generalized Predictive Control (GPC) by Clarke *et al.*(1987*a*, 1987*c*)are the most commonly mentioned ones. Although these variations of MPC use different process/noise model structures, performance criteria and optimization methodologies, all of them share some common features that can be summarized as follows.

- There is an explicit process model in step/impulse response, transfer function, state space or other form. In many formulations there is also an explicit or implicit noise model.
- Based on the process/noise model used, future outputs are predicted over a finite horizon called the prediction horizon.
- The predicted output has two distinct parts usually referred to as the 'free response' that depends on the information available at the current time and the 'forced response' that depends on the current and future control movements.
- An optimization problem is formulated that includes a user-specified performance criterion, with variable weighting (tuning) parameters, input-output constraints etc.

- The optimization problem is solved at each sampling instant to determine the present and future control moves.
- In accordance with the 'receding horizon principle', only the first control move is implemented and the entire calculation is repeated at the next control interval.

Therefore, in general, MPC is defined as a class of controllers that, using a process model, determines a set of manipulated variables by minimizing/maximizing some open-loop performance objectives over a finite time horizon from current to some extended future time. The most commonly used performance objective function is defined as

$$\begin{aligned} J_{MPC} &= \sum_{i=1}^{N_2} \left(r\left(k+i\right) - \hat{y}\left(k+i\right)\right)^T Q_i \left(r\left(k+i\right) - \hat{y}\left(k+i\right)\right) \\ &\quad + \sum_{i=1}^{M} u\left(k+i-1\right)^T R_i u\left(k+i-1\right) \\ &\quad + \sum_{i=1}^{M} \Delta u\left(k+i-1\right)^T \Lambda_i \Delta u\left(k+i-1\right) \end{aligned}$$

where $\hat{y}\left(k+i\right), i = 1, \cdots N_2$ are the predicted future outputs over a finite prediction horizon N_2, $r\left(k+i\right)$ are the future reference signals that are assumed to be known and $u\left(k+i-1\right)$, $\Delta u\left(k+i-1\right)$ are the present/future control and incremental control moves respectively that are determined by solving the quadratic optimization problem. In the MPC literature, M is called the control horizon, Q_i is the output weighting matrix, R_i is the input weighting matrix and Λ_i is known as the move suppression factor matrix. Other modifications and forms of the objective functions can be found in literature (Garcia and Morari 1982, Garcia *et al.* 1989, Cutler and Ramaker 1980, Eaton and Rawlings 1992, Morari and Lee 1999, Patwardhan 1999). The characterizing features of some commonly used MPC's such as DMC, MAC, GPC are discussed by Saudagar (1995). Garcia *et al.* (1989)specifically discuss the algorithm formulations of DMC and MAC. A detailed review of MPC basic principles and formulations can be found in (Garcia *et al.* 1989, De Keyser 1991, Eaton and Rawlings 1992, Patwardhan 1999).

Some of the advantages of MPC and reasons why MPC is so widely used in industrial applications are as follows (Camacho and Bordons 1999, Saudagar 1995,

Patwardhan 1999):

- MPC concepts such as those inherent in DMC are very intuitive and easy to understand even with little knowledge of advanced control theory. Therefore, it is easy to explain to industrial operators/technicians.
- MPC can handle large scale multivariable systems which are often difficult to handle by individual feedback controllers. It can be implemented on simple as well as complex systems.
- MPC can deal with hard constraints on inputs/outputs which are difficult to include in other control scheme implementations.
- Because the objective function is typically finite horizon and the optimization problem is linear or quadratic in nature, with the present day computational capabilities and powerful solution algorithms, it is not difficult to solve the optimization problem at every sampling instant.
- When the inequality constraints on the process inputs/outputs are inactive, the final control law is linear in nature and therefore easy to implement.
- With its flexible and open methodology, MPC can be extended and improved in many different ways. Recent developments include: optimization improvements using different algorithms, robust design and analysis, performance diagnosis, nonlinear modeling and control.

1.1.2 Historical Background and Present/Future Research Directions

The origin of predictive control or finite horizon control could be traced back to the 1960's by some researchers (*e.g.* (Garcia *et al.* 1989, Banerjee 1996)). The term *predictive control* has been found in the literature in as early as 1962 (*e.g.* (Horing 1962)) and the idea of long range predictive control (LRPC) can be traced back to 1964 (*e.g.* (Banerjee 1996, Kishi 1964)). However, extensive research and applications of model predictive control did not start until the late 1970's with the formulations and industrial applications of DMC (Cutler and Ramaker 1980), MAC/IDCOM (Richalet

et al. 1978, Rouhani and Mehra 1982), and the development of GPC by Clarke *et al.*(1987*a*, 1987*c*). It has been said that MPC was rediscovered in the 1980's and since then several books *e.g.* (Bitmead *et al.* 1990, Soeterboek 1992, Clarke 1994, Camacho and Bordons 1999) and hundreds of papers have been written on model-based predictive control.

In the 1990's several review papers (Garcia *et al.* 1989, Ricker 1991, Eaton and Rawlings 1992, Muske and Rawlings 1993, Lee 1996, Morari and Lee 1999) were published that discussed the theoretical developments of MPC and the likely future research directions. Step or impulse response model based DMC has become the most popular of the developed MPC's in industrial applications, particularly in the petro-chemical industries. On the other hand, the well known transfer function model based GPC has not been as popular as DMC to the industrial practitioners. Morari and Lee (1999), Saudagar (1995)have discussed some of the reasons behind this. Froisy (1994), Qin and Badgwell (1996)have reviewed the existing industrial MPC technologies along with their limitations and discussed likely future MPC variations from an industrial perspective. Lists of MPC applications in chemical and petroleum industries (*e.g.* distillation column, FCC, pulp and paper making, polymerization etc.) can be found in Ogunnaike and Ray (1994), Saudagar (1995)and Qin and Badgwell (1996). There are several commercial MPC packages (*e.g.* DMC, QDMC, IDCOMPC, HPC, HMPC etc.) developed by different companies. A list of them is available in publications by Froisy (1994)and Saudagar (1995).

Linear state space model based predictive controller design and analysis (*e.g.* (Li *et al.* 1989, Lee *et al.* 1994, Ricker 1990, Ricker 1991)) became very common in the late 1980's and early 1990's. Issues like feasibility of the solution to the constrained MPC problem, closed-loop stability, performance analysis and robustness of MPC were the main focus of MPC research in the 1990's and the beginning of the twenty-first century. Selection of a proper model, subspace model identification, nonlinear modeling and nonlinear MPC design also received a lot of attention during this period of time.

Researchers such as Qin and Badgwell (1996), Lee (1996), Morari and Lee (1999) pointed out that future research on MPC should include: identification improvement or model development in the context of MPC design, nonlinear modeling and con-

troller design, better use of uncertainty estimates, multiple performance objectives, performance monitoring and diagnosis, improved optimization and so on. Therefore, with the current interest of industrial practitioners in MPC, it can be said that MPC research is still open and will continue for several years to come. Closed-loop performance monitoring and/or diagnosis, robust stability design and analysis in the MPC context, non-linear MPC are key areas in which research and theoretical development scope are still open. MPC relevant modeling, integrated modeling, design & tuning also need improvement.

1.2 Research Objectives

1.2.1 Motivation

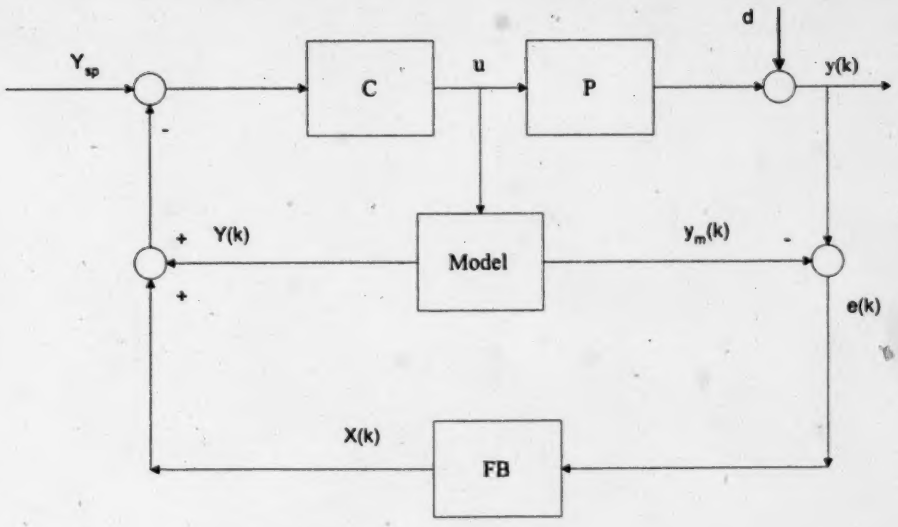
A classical MPC system is shown in Figure 1.1. The key components are:

1. **Model**
2. **Controller**
3. **Feedback** (estimates/predicts disturbances and MPM/uncertainty)

In more sophisticated MPC formulations, especially those based on MIMO state space models, the model and feedback components are combined into one larger module that performs

- state estimation
- output and disturbance estimation
- noise/disturbance filtering
- parameter estimation etc.

These (state space based) techniques are very powerful but the MPC structure is not as transparent to the user, *e.g.* the model/prediction and (disturbance) feedback paths are not separated as in Figure 1.1. This "separation" is practically helpful to application-oriented personnel in the *process control area* who want to separate the "servo" and "regulatory" control functions.



$$Y(k) = [\ \hat{y}(k) + \hat{y}(k+1) + \cdots + \hat{y}(k+p)\ ]^T$$

$$X(k) = [\ \hat{x}(k) + \hat{x}(k+1) + \cdots + \hat{x}(k+p)\ ]^T$$

P = Physical process or system to be controlled

FB = Feedback

Figure 1.1: Classical MPC Structure

For practical applications the basic/classical MPC system shown in Figure 1.1 is augmented as indicated by Figure 1.2 to include

1. model **identification** and parameter estimation

2. **performance assessment** to compare the *achieved* control with the "best possible" control and/or more practical LQG or MPC benchmarks.

3. system monitoring and on-line **tuning** (where "system" includes all components of the complete MPC system) is required to adapt to changes in the physical process or in the performance of individual modules. For example, if the identified model is 'poor' or if the feedback prediction $X(k)$ is unreliable then the controller should be redesigned/retuned so that it is less aggressive and more robust.

4. **data-filtering/preprocessing** that can range from simple noise filters to sophisticated MIMO analysis/correlation etc. techniques.

5. higher-level functions such as **process optimizers** are used in many industrial applications to generate set-point or target trajectories and can produce significant economic benefits.

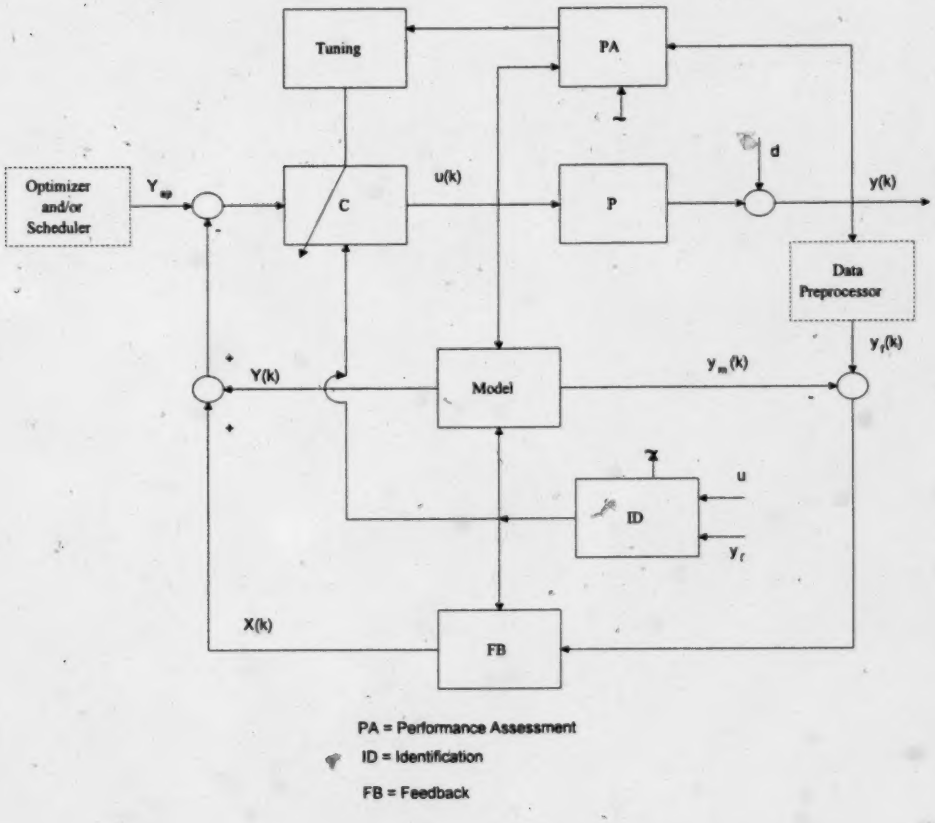


Figure 1.2: MPC System Supervisor

There are multiple methodologies and design approaches for each component in Figure 1.2. Therefore it is not difficult to propose a "new" MPC system using different design and implementation strategies for each component. *The challenge is to select and design each MPC component so that they integrate well with the other MPC components and facilitate specification and on-line tuning of overall MPC-system per-*

formance. With this "challenge" in mind the objectives of this thesis were developed as outlined in the next subsection.

1.2.2 General Objectives

The overall objective of this thesis was to develop a "complete" MPC system that incorporated the following characteristics.

1. A "dual-model" that combined the flexibility and ease of interpretation of step (or FIR) models used in DMC with the "parametric-efficiency" and computational advantages of parametric models such as those used in GPC and state space methods.

2. An integrated approach to the selection and formulation of each component so that they work well together and can be designed *a priori* and tuned on-line to accommodate changes in the process and/or performance objectives.

3. A common basis for all the MPC components that facilitates design and analysis using modern control theory and computational packages such as MATLAB.

4. A complete MPC system that would meet the practical requirements of industrial applications operated by plant personnel.

1.3 Contents and Contributions of the Thesis

Not surprisingly the key distinguishing features of any **Model**-based Predictive Control system is the selected **model** structure. The MPC system developed in this thesis uses a "dual-model" that combines a user-specified number of impulse (or FIR or Markov) parameters to define the time-delay and initial portion of the process response together with a parametric (ARIMA) model to define the balance (slow dynamics) of the process response. This is illustrated in Figure 1.3.

The μ-point step (impulse, FIR or Markov) model of the initial process response means that it is NOT necessary to specify, *a priori*, the time delay, model structure etc. as required for parametric models. The slow or residual response from $k+$

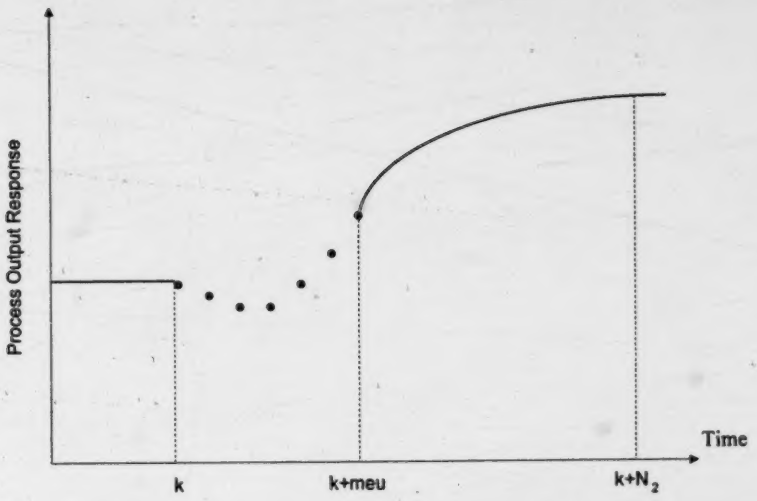


Figure 1.3: Dual-model Representation of a Process Step Response

μ to $k + N_2$ (N_2 =MPC prediction horizon) is adequately represented by a low-order, parametric model as is common in the process industries. Three points are particularly important about the "dual-model" used in this thesis:

1. The MPC system developed using the dual-ARMarkov model is shown to be mathematically equivalent to DMC as $\mu \rightarrow N_2$ and to GPC as $\mu \rightarrow 1$. Thus by specifying $1 \leq \mu \leq N_2$ the user can obtain any desired combination of DMC (non-parametric) and GPC (parametric) characteristics.

2. The dual-ARMarkov model provides a sound theoretical basis for design, analysis and simulation and is shown to have advantages for use in other MPC components, including identification, control, performance assessment and tuning.

3. A separate "disturbance term" incorporated into the dual-ARMarkov model provides a basis for more accurate prediction of the process output and leads to a mechanism for independent tuning for 'servo' and 'regulatory' control.

The main areas studied in this thesis are identified in Figure 1.2 (solid boxes) and in column 1 of Table 1.1. The details of the work done in this thesis project are

contained in the Chapters listed in column 2 of Table 1.1 and summaries of the key contributions are in the sections listed in column 3.

However, very briefly the contributions of this thesis are:

1. **Model:** a dual-ARMarkov model as discussed above

2. **Identification:** a multivariable, ARMarkov identification procedure that uses conventional least-squares to directly determine the μ Markov parameters plus the order and parameters of the "residual response model".

3. **Controller:** provides any desired combination of DMC (conservative) and GPC (aggressive) characteristics together with a clear separation of 'servo' and 'regulatory' control.

4. **Feedback:** the dual ARMarkov model permits separation of the model-based feedback predictions from the "disturbance" prediction analogous to $Y(k)$ and $X(k)$ in Figure 1.1.

5. **Performance:** overall control performance is evaluated using familiar time-domain metrics such as rise time (dominant time constant), theoretical metrics such as sensitivity functions and/or relative to calculated (e.g. LQG) performance objectives.

6. **Tuning:** a single parameter, $0 \leq \alpha \leq 1$, provides *stable, robust* control over the full range of performance and robustness specified *a priori* by the user during the design phase.

Table 1.1: Overview of Thesis Work on MPC

MPC Component	Chapter(s) with details	Section with Summary
(1) Model	2, 3, 4, (9)	12.1, point #1
(2) Identification	2, 3, 5	12.1, point #2
(3) Controller	6, 8, 9	12.1, point #3
(4) Feedback	7	12.1, point #4
(5) Performance/robustness	10	12.1, point #5
(6) On-line tuning	11	12.1, point #6

Part I

ARMarkov Identification

This part of the thesis focuses on the ARMarkov models that are used later in the thesis for development of Model-based Predictive Controllers (MPC). As shown in Chapter 2 the ARMarkov model is a combination of non-parametric and parametric input-output models and does not require accurate values of the process time delay or actual model order. At one extreme the ARMarkov model consists only of non-parametric terms and is equivalent to the model used for DMC (Cutler and Ramaker 1980). At the other extreme it reduces to a parametric model identical to the model used for GPC (Clarke *et al.* 1987*b*).

In Chapters 2 and 3 the existing identification method for ARMarkov models of SISO systems is extended to the multivariable case. The result is a straightforward linear regression algorithm that performs well in practical applications. The statistical analysis in Chapter 4 shows advantages such as better consistency, variance/covariance and confidence bounds of the estimated parameters compared to the same properties of the parameters estimated by other linear regression methods such as FIR or ARX. Chapter 5 shows how the interactor (time-delay) matrix can be obtained directly from the estimated Markov parameters.

MPC design and on-line tuning are presented in Parts II and III respectively.

Chapter 2

Estimation of Markov Parameters using ARMarkov Identification[1]

2.1 Introduction

System identification is a well known research area in the control community. It is important in both academia and industry. System identification is the process of developing or updating a mathematical representation of a process using experimental data. There are several reasons for developing a mathematical model such as to understand the process dynamics, to design a controller, to implement control-loop performance monitoring/assessment, train the plant operator and so on. When any input is introduced to a process, outputs from the process are obtained that depend on the dynamic behavior of the process. A model is a mathematical function that maps the process input(s) to the process output(s). The dynamic behavior of the process can be estimated in the form of mathematical models by analyzing the process input(s) and output(s).

There are different methods of identification and different models have been used for different processes. Since the mid-sixties, several techniques have been developed for system identification and many textbooks have been written focusing on different techniques and different types of models. Most of the practical chemical processes are nonlinear in nature. However, linear models are used in most applications because linear models are easier to identify and analyze. The most commonly used linear models are: time domain, state space and frequency domain models. All the linear

[1]This chapter is a part of the paper published in the *Chemical Engineering Science*, May 2000. It was also presented at the 1998 CSCHE Conference, London, Ontario, Canada.

models are equivalent and can be converted from one form to the other. However, each form has some advantages over the other depending on the application. The time domain models are easy to explain to industrial operators whereas, state space models can be easily used for multivariable systems and theoretical analysis is also more convenient for multivariable systems.

There are mainly two types of time domain or transfer function models, non-parametric and parametric models. Non-parametric models are higher order, but require less specific a-priori information on the process. On the other hand, low order parametric models require a-priori information such as time delay specification, model order etc. of the process.

The least-squares identification method along with a linear regression model is one of the easiest and most commonly used approaches. The use of the least-squares method can be traced back to Gauss (1809). Among the commonly used least-squares methods are the FIR (Finite Impulse Response), ARX (Auto Regressive exogenous) etc. The main advantage of the least-squares method is that the global minimum can be found efficiently (no local minimum exists) (Ljung 1987). The main disadvantage is that if the noise term in the linear regression model is not white noise, the estimated parameters will not converge asymptotically to the actual parameters. The FIR and ARX methods will be discussed further in Section 2.2.

Most Model Predictive Controllers (MPC) need step response or impulse response coefficients of the process (directly or indirectly). In most of the MPC's, the 'dynamic matrix' is constructed using the first few step/impulse response coefficients. It is important to accurately estimate the step/impulse response (or Markov parameter) which represent the fast or initial portion of the process response. If these Markov parameters can be determined accurately, performance can be improved. Plants are often high order, but they are approximated by low-order models for the purpose of control design. In such cases, the lower order approximation must be reliable, at least for the fast dynamics part of the step/impulse response.

Time delay estimation plays a very important role in control system design and in multivariable performance assessment. In multivariate cases, the time delays are defined by a polynomial matrix called the interactor matrix. Huang (1997) proposed a method of determining the interactor matrix that does not require a priori infor-

mation on the process transfer functions and uses a linear combination of the process Markov parameters. He also noted that better interactor estimation is obtained when the Markov parameters are estimated directly rather than transforming process transfer functions into Markov parameters. The first step in his method of estimating the interactor matrix is to determine the Markov parameters of the multivariable system. Huang (1997) used correlation analysis to estimate the Markov parameters of MIMO systems. However, in the presence of non-white disturbances, the Markov parameters estimated by correlation analysis are significantly affected by the disturbances (Soderstrom and Stoica 1989).

Process dynamics such as non-linearities, abnormal step/impulse behavior in the fast dynamics cannot be captured well by a linear model. Therefore, direct estimation rather than converting input-output or state space models to Markov parameters is important for both interactor matrix estimation (Huang and Shah 1999) as well as for MPC design since they use the Markov parameters directly. Non-parametric methods such as the FIR method estimate the Markov parameters directly. However, non-parametric methods need very high model order to capture the full dynamics of the process. On the other hand, parametric models are highly affected by disturbances and if inaccurate disturbance models or no disturbance models are considered, then the Model Plant Mismatch (MPM) becomes very high.

A better method for the determination of the impulse response or the Markov parameters is proposed in this chapter. It is called the ARMarkov/LS method as introduced by Hyland (1991) and used by Akers and Bernstein(1997, 1999). This procedure uses a least-squares algorithm with the ARMarkov representation of a process model which relates the present output(s) with past outputs and past inputs. The ARMarkov model has the same form as an ARX model except that it explicitly contains more than one Markov parameter. Therefore, the ARMarkov model can be viewed as an overparameterized and structurally constrained ARX representation (Akers and Bernstein 1997). In other words, the ARMarkov model is a combination of a non-parametric model and a parametric residual model. The ARMarkov/LS identification method uses a regressor vector consisting of past input-output data with a least-squares criterion to estimate a weighting matrix consisting of a user specified number of Markov parameters. The proposed method assures the reliabil-

ity of the step/impulse response coefficients for the fast dynamics part. Analysis of the statistical properties of the estimated ARMarkov model parameters, shown later in Chapter 4, proves that the Markov parameters obtained by using the AR-Markov method are consistent even in the presence of colored disturbances. Hyland (1991) has shown that the ARMarkov model representation is less sensitive to noisy measurements than the ARX or ARMA model representations. Venugopal and Bernstein (1997*a*) noted that perturbations to ARMarkov parameters have less impact on model behavior as compared to ARMA representations. An ARMarkov adaptive control (AAC) algorithm developed by Venugopal and Bernstein (1997*b*) was used by Venugopal and Bernstein(1997*a*, 1997*b*, 2000)for disturbance suppression purposes. Sane *et al.* (1999*a*) analyzed properties such as robustness of the AAC. Sane *et al.* (1999*b*) studied the behavior of the AAC under simultaneous identification and noted that the algorithm neither requires nor utilizes a model of the feedback transfer function. Moreover, a recursive algorithm can be used to update the parameter estimates that saves computational cost due to the overparameterized model representation.

An introduction to linear regression models and least-squares methods is included in Section 2.2. Section 2.3 describes the ARMarkov model representation for SISO systems and the LS identification of the model parameters. Extension of the ARMarkov method to MIMO systems and a recursive algorithm for the ARMarkov least-squares method are discussed in Section 2.4 and Section 2.5 respectively. Applications of the ARMarkov method are given in Section 2.6.

2.2 Linear Regression Models and Least-squares Methods

'Regression analysis is the statistical methodology for predicting values of one or more response (dependent) variables from a collection of predictor (independent) variable values'(Johnson and Wichern 1988). As mentioned earlier, a linear regression model is the simplest model structure and a simple method such as least-squares can be used for the determination of the model parameters. In linear regression, the output predictor is a scalar product between a data vector, ϕ and the parameter vector, θ. The known data vector, ϕ is called the *regressor* vector. The classical linear regression

model is defined as

$$\begin{aligned} y &= \phi^T\theta + \epsilon \\ E(\epsilon) &= 0, \text{ and } Covariance(\epsilon) = \sigma^2 I \end{aligned}$$

where θ and σ^2 are unknown parameters. In discrete time systems, the predictor is defined as,

$$\widehat{y}(k) = \phi^T(k)\theta \tag{2.1}$$

and the prediction error becomes

$$\begin{aligned} \epsilon(k) &= y(k) - \widehat{y}(k) \\ &= y(k) - \phi^T(k)\theta \end{aligned}$$

The least-squares criterion is defined as

$$\min_{\theta} J\left(Z^N, \theta\right) = \min_{\theta} \frac{1}{N}\sum_{k=1}^{N}\frac{1}{2}\left[y(k) - \phi^T(k)\theta\right]^2$$

and the least-squares estimate (LSE) of the parameters is

$$\begin{aligned} \hat{\theta}_N^{LS} &= \arg\min_{\theta} J\left(Z^N, \theta\right) \\ &= \left[\frac{1}{N}\sum_{k=1}^{N}\phi(k)\phi^T(k)\right]^{-1}\frac{1}{N}\sum_{k=1}^{N}\phi(k)y(k) \end{aligned}$$

2.2.1 ARX Model

The ARX model structure is, arguably the easiest linear regression parametric model to use for system identification. It uses the following input-output difference equation to formulate the model.

$$\begin{aligned} y(k) + a_1 y(k-1) + \cdots + a_{na} y(k-n) &= b_0 u(k) + b_1 u(k-1) + \cdots \\ &\quad + b_{nb} u(k-n) + \epsilon(k) \qquad (2.2) \\ \text{or } A\left(q^{-1}\right) y(k) &= B\left(q^{-1}\right) u(k) + \epsilon(k) \qquad (2.3) \end{aligned}$$

where $A\left(q^{-1}\right) y(k)$ is referred to as autoregressive (AR) part and $B\left(q^{-1}\right) u(k)$ as exogenous (X) input part. To see the ARX model as a linear regression model,

compare equation (2.1) with equation (2.2) or (2.3). The regressor vector consists of input-output sequence as

$$\phi(k) = [-y(k-1), \cdots, -y(k-na), u(k), \cdots, u(k-nb)]^T$$

and the parameter vector is defined as

$$\theta = [a_1, \cdots, a_{na}, b_1, \cdots b_{nb}]^T$$

The parameter vector is estimated using the least-squares method.

2.2.2 Finite Impulse Response (FIR) Model

A Finite Impulse Response (FIR) Model is a linear, time-invariant (LTI), non-parametric model structure. A LTI model can be defined by the full impulse responses of the process. Non-parametric models, e.g. models used in correlation analysis and/or step/impulse response analysis, do not explicitly employ a finite dimensional parameter vector. The FIR model, also known as a 'truncated weighting function' model is a simplified form of the LTI model

$$y(k) = \sum_{i=0}^{\infty} h_i u(k-i) + \epsilon(k)$$

used in correlation analysis where f_i are the impulse response coefficients or Markov parameters of the process. Assuming

$$h_i = 0 \quad \text{for } i \geq M$$

the FIR or 'truncated weighting function' model is written as

$$y(k) = h_0 u(k) + h_1 u(k-1) + \cdots + h_{M-1} u(k-M+1) + \epsilon(k) \tag{2.4}$$

The number of Markov parameters f_i i.e. M should be large enough, compared to the time constant of the process, to avoid truncation error and to get a good approximation of the infinite dimensional model structure. The FIR model can also be obtained when $na = 0$ in (2.2). The linear regression form of the model is obtained using a regressor vector consisting of present/past inputs only i.e.

$$\phi(k) = [u(k), \cdots, u(k-nb)]^T$$

and

$$\begin{aligned} \theta &= [b_1, \cdots b_{nb}]^T \\ &\equiv [h_0, \cdots h_{M-1}]^T \end{aligned}$$

The Markov parameters or impulse response coefficients for this model can be estimated by using LS identification or by using correlation analysis.

2.3 The ARMarkov Method

Let $G(z) \triangleq \left[\begin{array}{c|c} A_x & B_x \\ \hline C_x & D_x \end{array}\right]$ denote the transfer function of a discrete, linear, time-invariant SISO system having the state space form

$$x(k+1) = A_x x(k) + B_x u(k), \tag{2.5}$$

$$y(k) = C_x x(k) + D_x u(k) \tag{2.6}$$

where $A_x \in R^{n \times n}, B_x \in R^{n \times 1}, C_x \in R^{1 \times n}$, and $D_x \in R$.

The Markov parameters h_j are defined by (Akers and Bernstein (1997)) as

$$\begin{aligned} h_j &\triangleq D_x \qquad \text{for } j = 0 \\ &\triangleq C_x A_x^{j-1} B_x \qquad \text{for } j \geq 1 \end{aligned} \tag{2.7}$$

The transfer function of equations (2.5) and (2.6) is

$$G(z) \triangleq C_x(zI - A_x)^{-1} B_x + D_x \tag{2.8}$$

The Markov parameter representation of $G(z)$ is

$$G(z) = \sum_{j=0}^{\infty} h_j z^{-j} \tag{2.9}$$

The ARX transfer function representation of $G(z)$ is

$$G_{ARX}(z) = \frac{b_0 z^n + b_1 z^{n-1} + \cdots + b_n}{z^n + a_1 z^{n-1} + \cdots + a_n} \tag{2.10}$$

where

$$det\,(zI - A_z) = z^n + a_1 z^{n-1} + \cdots + a_n$$
$$\text{and} \quad b_i \in \Re^{l \times m},\ i = 0, \cdots, n$$

Equating the transfer functions in equations (2.9) and (2.10) and then multiplying both sides by $z^n + a_1 z^{n-1} + \cdots + a_n$ results in the relationship

$$\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} h_0 & 0 & \cdots & 0 \\ h_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_n & \cdots & h_1 & h_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} \tag{2.11}$$

$$h_{n+j} = -\sum_{i=1}^{n} a_i h_{n+j-i}, \qquad \text{for } j > 0 \tag{2.12}$$

The ARX transfer function representation in (2.10) contains the first Markov parameter of the process i.e. $h_0 = b_0$.

The goal here is to blend the two transfer functions i.e. the non-parametric representation in (2.9) and the parametric representation in (2.10), to incorporate more than one Markov parameter and maintain the rationality of the transfer function. The time-domain representation of the ARX model is

$$y(k) = -a_1 y(k-1) - \cdots - a_n y(k-n) + b_0 u(k) + \cdots + b_n u(k-n), \text{ for } k \geq 0 \tag{2.13}$$

Now substituting k by $k-1$ in (2.13) yields

$$y(k-1) = -a_1 y(k-2) - \cdots - a_n y(k-n-1) + b_0 u(k-1) + \cdots + b_n u(k-n-1),$$
$$\text{for } k \geq 0 \tag{2.14}$$

and using (2.14) to replace the value of $y(k-1)$ in (2.13) results in

$$\begin{aligned} y(k) = {} & \left(a_1^2 - a_2\right) y(k-2) + \left(a_1 a_2 - a_3\right) y(k-3) + \cdots + \left(a_1 a_{n-1} - a_n\right) y(k-n) \\ & + a_1 a_n y(k-n-1) + b_0 u(k) + \left(b_1 - a_1 b_0\right) u(k-1) + \left(b_2 - a_1 b_1\right) u(k-2) \\ & + \cdots + \left(b_n - a_1 b_{n-1}\right) u(k-n) - a_1 b_n u(k-n-1) \end{aligned} \tag{2.15}$$

Using (2.11), the first two Markov parameters are: $h_0 = b_0$ and $h_1 = b_1 - a_1 b_0$. Then (2.15) can be written as

$$\begin{aligned} y(k) = {} & \alpha_1 y(k-2) + \alpha_2 y(k-3) + \cdots + \alpha_n y(k-n-1) \\ & + h_0 u(k) + h_1 u(k-1) + \beta_1 u(k-2) + \cdots + \beta_n u(k-n-1) \end{aligned} \tag{2.16}$$

where $\alpha_1, \cdots, \alpha_n \in \Re$ and $\beta_1, \cdots, \beta_n \in \Re$ are defined as

$$\begin{aligned}
\alpha_i &= a_1 a_i - a_{i+1} \quad i = 1, \cdots, n-1 \\
\alpha_n &= a_1 a_n \\
\beta_i &= b_{i+1} - a_1 b_i \quad i = 1, \cdots, n-1 \\
\beta_n &= a_1 b_n
\end{aligned}$$

Equation (2.16) explicitly contains first two Markov parameters, h_0 and h_1 corresponding to transfer function $G(z)$. Note that the input-output model in (2.16) is a combination of the ARX model in (2.2) and the FIR model in (2.4). Since this transfer function is a blend of the *ARX* transfer function and the transfer function representing the *Markov* parameters or impulse response coefficients i.e. the FIR transfer function, the blended transfer function is referred to as the *ARMarkov* transfer function.

Repeating the above sequence $(\mu - 1)$ times, produces the following ARMarkov representation corresponding to $G(z)$ with μ Markov parameters

$$G_{ARMark}(z) = \frac{h_0 z^{\mu+n-1} + \cdots + h_{\mu-1} z^n + \beta_{\mu,1} z^{n-1} + \cdots + \beta_{\mu,n}}{z^{\mu+n-1} + \alpha_{\mu,1} z^{n-1} + \cdots + \alpha_{\mu,n}} \tag{2.17}$$

The ARMarkov time-domain representation is given by

$$y(k) = -\sum_{j=1}^{n} \alpha_j y(k-\mu-j+1) + \sum_{j=1}^{\mu} h_{j-1} u(k-j+1) + \sum_{j=1}^{n} \beta_j u(k-\mu-j+1) \tag{2.18}$$

and involves only the first μ Markov parameters $h_{0,\cdots}, h_{\mu-1}$. The parameters $\alpha_1, \cdots, \alpha_n \in R^{1\times n}$ are functions of the ARX coefficients and the Markov parameters (Akers and Bernstein 1997). Here n is the order of the parametric residual model and is called the order of the ARMarkov model. The ARX time-domain representation is a special form of the ARMarkov time-domain representation with $\mu = 1$.

2.3.1 ARMarkov/LS Identification Algorithm

The ARMarkov regressor vector, ϕ_μ for equation (2.18) is written as

$$\phi_\mu(k) \triangleq \left[\; y(k-\mu) \;\; \cdots \;\; y(k-\mu-n+1) \;\; u(k) \;\; \cdots \;\; u(k-\mu-n+1) \;\right]^T \tag{2.19}$$

The process output can be expressed as

$$y(k) = \mathbf{W}_\mu \boldsymbol{\phi}_\mu(k), \tag{2.20}$$

where the weighting matrix is given by,

$$\mathbf{W}_\mu = [-\mathbf{A}_\mu \quad h_0 \quad \cdots \quad h_{\mu-1} \quad \mathbf{B}_\mu] \tag{2.21}$$

$$\mathbf{A}_\mu = [\alpha_1, \cdots, \alpha_n] \in R^{1\times n} \tag{2.22}$$

$$\mathbf{B}_\mu = [\beta_1 \ \cdots \ \beta_n] \in R^{1\times n} \tag{2.23}$$

Let $\hat{\mathbf{W}}_\mu$ be the estimate of the weighting matrix and $\hat{y}(k)$ be the estimated output. Then

$$\hat{y}(k) = \hat{\mathbf{W}}_\mu \boldsymbol{\phi}_\mu(k) \tag{2.24}$$

Define the output prediction error, $\varepsilon(k)$ as

$$\varepsilon(k) = y(k) - \hat{y}(k) \tag{2.25}$$

and the cost function, J, in terms of the output error as

$$J = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{2}\varepsilon^2(k) \tag{2.26}$$

$$= \frac{1}{2N}\sum_{k=1}^{N}\Big(y(k) - \hat{\mathbf{W}}_\mu \boldsymbol{\phi}_\mu(k)\Big)^2 \tag{2.27}$$

where N is the number of total data points. Then $\hat{\mathbf{W}}_\mu$ is a strict minimization of J iff

$$\hat{\mathbf{W}}_\mu = \left[\frac{1}{N}\sum_{k=1}^{N} y(k)\boldsymbol{\phi}_\mu^T(k)\right]\left[\frac{1}{N}\sum_{k=1}^{N}\boldsymbol{\phi}_\mu(k)\boldsymbol{\phi}_\mu^T(k)\right]^{-1} \tag{2.28}$$

Matrix $\frac{1}{N}\sum_{k=1}^{N}\boldsymbol{\phi}_\mu(k)\boldsymbol{\phi}_\mu^T(k)$ contains the covariance estimates of $u(k)$ and $y(k)$ and must be non-singular for the inverse to exist. When $\mu = 1$, equation (2.28) is identical to the ARX/LS identification algorithm.

2.4 Extension of the ARMarkov Method to MIMO Systems

Akers and Bernstein (1997) suggested that for MIMO systems, each individual input-output pair be considered separately to generate the corresponding Markov parameters. The estimated parameters could then be stacked in matrix form. However, as shown below, the ARMarkov/LS method can be reformulated for MIMO systems to estimate the Markov parameter blocks directly. The input-output relationship is expressed as,

$$\mathbf{Y}(k) = -\sum_{j=1}^{n} \boldsymbol{\alpha}_j \mathbf{Y}(k-\mu-j+1) + \sum_{j=1}^{\mu} \mathbf{H}_{j-1} \mathbf{U}(k-j+1) + \sum_{j=1}^{n} \boldsymbol{\beta}_j \mathbf{U}(k-\mu-j+1) \tag{2.29}$$

where $\mathbf{Y}(k) = [y_1(k),\, y_2(k) \cdots y_l(k)]^T$, $\mathbf{U}(k) = [u_1(k),\, u_2(k) \cdots u_m(k)]^T$ and $\boldsymbol{\alpha}$, $\mathbf{H}_j$ & $\boldsymbol{\beta}$ are of appropriate sizes. The number of outputs and inputs are l and m respectively. The regressor vector is expressed as

$$\Phi_\mu(k) \triangleq \left[\, \mathbf{Y}(k-\mu)^T \;\; \cdots \;\; \mathbf{Y}(k-\mu-n+1)^T \;\; \mathbf{U}(k)^T \;\; \cdots \;\; \mathbf{U}(k-\mu-n+1)^T \,\right]^T \tag{2.30}$$

Following the same least-squares procedure used for the SISO systems, the cost function becomes

$$J = \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \tag{2.31}$$

where $\boldsymbol{\epsilon}$ is the output error vector and the estimate of the weighting matrix, $\hat{\mathbf{W}}_\mu$ is

$$\hat{\mathbf{W}}_\mu = \left[\frac{1}{N} \sum_{k=1}^{N} \mathbf{Y}(k) \Phi^T(k)\right] \left[\frac{1}{N} \sum_{k=1}^{N} \Phi(k) \Phi^T(k)\right]^{-1} \tag{2.32}$$

where

$$\hat{\mathbf{W}}_\mu = \left[-\hat{\mathbf{A}}_\mu \quad \hat{\mathbf{H}}_{-1} \quad \cdots \quad \hat{\mathbf{H}}_{\mu-2} \quad \hat{\mathbf{B}}_\mu\right], \tag{2.33}$$

$$\hat{\mathbf{A}}_\mu = [\hat{\boldsymbol{\alpha}}_{\mu,1},\, \cdots \hat{\boldsymbol{\alpha}}_{\mu,n}] \in R^{l \times nl},\; \hat{\mathbf{H}}_j \in R^{l \times m},$$

$$\hat{\mathbf{B}}_\mu = \left[\hat{\boldsymbol{\beta}}_{\mu,1} \quad \cdots \quad \hat{\boldsymbol{\beta}}_{\mu,n}\right] \in R^{l \times nm} \tag{2.34}$$

2.5 Recursive ARMarkov/LS Identification

In order to support on-line decisions such as performance monitoring/assessment and on-line control, in most applications, the process model must be updated on-line. This updating procedure is known as recursive identification. The ARMarkov identification technique can be formulated in recursive form as described in the following sections.

2.5.1 SISO Systems

The least-squares estimate of the weighting matrix which contains the process parameters is defined in equation (2.28). This expression can be written in a recursive way as described by Ljung and Soderstrom (1983) as

$$\hat{\mathbf{W}}_\mu(t) = \left[\frac{1}{t}\sum_{k=1}^{t} y(k)\Phi^T(k)\right]\left[\frac{1}{t}\sum_{k=1}^{t}\Phi(k)\Phi^T(k)\right]^{-1} \tag{2.35}$$

where t represents the current time instant. Following the derivation of the *recursive least squares* (RLS) *algorithm* in Ljung and Soderstrom (1983) and applying it to the ARMarkov identification, the recursive estimate of the weighting matrix can be written as

$$\hat{\mathbf{W}}_\mu(t) = \hat{\mathbf{W}}_\mu(t-1) + L(t)\left[y(t) - \hat{\mathbf{W}}_\mu^T(t-1)\Phi(t)\right]$$

$$\mathbf{L}(t) = \frac{\mathbf{P}(t-1)\Phi(t)}{\frac{1}{\alpha_t} + \Phi^T(t)\mathbf{P}(t-1)\Phi(t)} \tag{2.36}$$

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \frac{\mathbf{P}(t-1)\Phi(t)\Phi^T(t)\mathbf{P}(t-1)}{\frac{1}{\alpha_t} + \Phi^T(t)\mathbf{P}(t-1)\Phi(t)}$$

where $\alpha_t = \frac{1}{2}$ is the coefficient in the cost function of the output error as described in equation (2.26).

The initial values for equations (2.10) through (2.15) can be obtained as in Ljung and Soderstrom (1983). If the starting time is t_0,

$$\begin{aligned} P(t_0) &= \left[\sum_{k=1}^{t_0}\alpha_k\Phi(k)\Phi^T(k)\right]^{-1}, \\ \hat{\mathbf{W}}_\mu(t_0) &= P(t_0)\left[\sum_{k=1}^{t_0}\alpha_k y(k)\Phi^T(k)\right] \end{aligned} \tag{2.37}$$

and if the recursion starts at time $t = 0$, the initial values can be started with some invertible matrix $\mathbf{P}(0)$ and a vector $\hat{\mathbf{W}}_\mu(0)$.

2.5.2 MIMO Systems

For MIMO systems, the recursive estimation is the same as for the case of SISO systems described in the previous section except that the regressor vector, $\Phi(t)$ is defined as in equation (2.30) instead of in equation (2.19) and the estimate of the weighting matrix, $\hat{\mathbf{W}}_\mu(t)$ is matrix as in equation (2.32) rather than a vector as in equation (10.3).

2.6 Simulation Examples

This section describes simulations carried out to estimate the Markov parameters from process input-output data.

2.6.1 Open-Loop SISO Systems

Example 2.1

Consider the third order SISO system with the transfer function,

$$G(z) = \frac{0.0077z^{-1} + 0.0212z^{-2} + 0.0036z^{-3}}{1 - 1.9031z^{-1} + 1.1514z^{-2} - 0.2158z^{-3}} \tag{2.38}$$

subjected to colored noise generated by passing white noise through the disturbance transfer function

$$H(z) = \frac{1}{1 - 0.95z^{-1}} \tag{2.39}$$

Since the ARMarkov method involves simple linear regression, the estimated Markov parameters are compared with the Markov parameters determined by correlation analysis (or FIR) and the ARX method which also use linear regression. The ARMarkov method gives the best estimate of the process impulse response (Markov parameters) as shown in Figure 2.1. The Markov parameters determined by the other methods have much larger errors because these methods try to fit the process over the

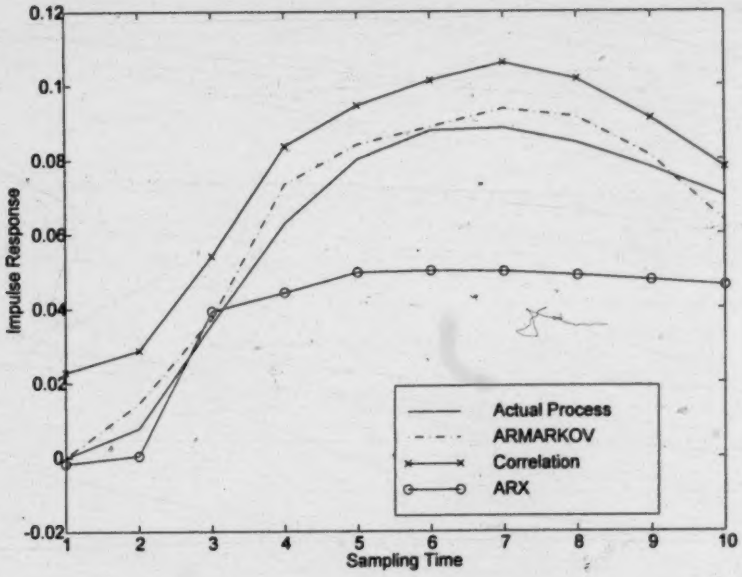


Figure 2.1: Impulse Response of a Third Order System

whole range. The ARMarkov method gives better estimates of the first few Markov parameters which are needed to determine the process delay/interactor matrix or to design Model-based Predictive Controllers.

2.6.2 Open-Loop MIMO Systems

Example 2.2

This example estimates the Markov parameters of a process which has different delays and different orders in the input-output pairs of the transfer function matrix. The elements of the diagonal disturbance transfer function matrix approximate random step-type disturbances which are common in chemical processes.

Consider a 2×2 multivariable system with the transfer function matrix

$$G_p(z) = \begin{bmatrix} \frac{.1091z^{-1}+0.07z^{-2}}{1-1.0844z^{-1}+0.2636z^{-2}} & \frac{0.0406z^{-1}-0.0113z^{-2}+0.0009z^{-3}}{1-0.6495z^{-1}+0.0482z^{-2}} \\ \frac{0.0406z^{-1}-0.0299z^{-2}-0.0047z^{-3}}{1-0.7849z^{-1}+0.7902z^{-2}} & \frac{0.0077z^{-2}+0.0212z^{-3}+0.0036z^{-4}}{1-1.9031z^{-1}+1.1514z^{-2}-0.2158z^{-3}} \end{bmatrix} \tag{2.40}$$

and the disturbance transfer function matrix

$$H = \begin{bmatrix} \frac{1}{1-0.95z^{-1}} & 0 \\ 0 & \frac{1}{1-0.99z^{-1}} \end{bmatrix} \tag{2.41}$$

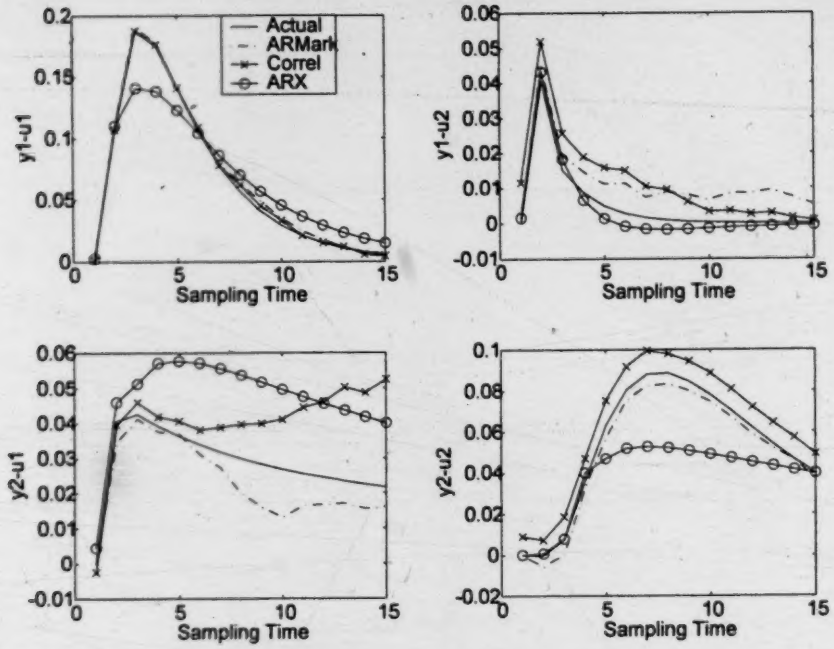


Figure 2.2: Impulse Response of a 2x2 MIMO System.

From the impulse response plots in Figure 2.2, it is obvious that the Markov parameters determined by the ARMarkov method are very close to the Markov parameters of the actual plant and better than the Markov parameters determined by the ARX and the Correlation methods (FIR). The first and second Markov parameter blocks determined by different methods are given in Table 2.1.

2.7 Model Identification of a Pilot Scale CSTH Using Experimental Data

The proposed ARMarkov identification method was evaluated on the pilot scale Continuous Stirred Tank Heater (CSTH) in the Computer Process Control (CPC) laboratory in the Chemical & Materials Engineering Department at the University of Alberta. A schematic diagram of the process is shown in 2.3. The CSTH consists of a cylindrical tank with an exit pipe and a steam coil running through the tank. There are two valves to manipulate the inlet cold water and steam flow rates. Another valve

Table 2.1: Markov Parameter blocks of a 2x2 MIMO system

Method	go	$g1$
Actual	$\begin{bmatrix} 0.1091 & 0.0406 \\ 0.0406 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.1883 & 0.0151 \\ 0.0426 & 0.0077 \end{bmatrix}$
ARMARKOV	$\begin{bmatrix} 0.1063 & 0.0429 \\ 0.0410 & 0.0060 \end{bmatrix}$	$\begin{bmatrix} 0.1844 & 0.0143 \\ 0.0376 & 0.0035 \end{bmatrix}$
FIR	$\begin{bmatrix} 0.1098 & 0.0476 \\ 0.0246 & -0.0100 \end{bmatrix}$	$\begin{bmatrix} 0.1866 & 0.0194 \\ 0.0201 & -0.0099 \end{bmatrix}$
ARX	$\begin{bmatrix} 0.1451 & 0.0437 \\ 0.0447 & 0.0032 \end{bmatrix}$	$\begin{bmatrix} 0.1557 & 0.0166 \\ 0.0385 & -0.0036 \end{bmatrix}$

at the exit pipe can be adjusted manually. A number of thermocouples are placed along the exit pipe to produce different time delays in the temperature measurement. A DP cell at the bottom of the tank measures the level and an electromagnetic flow meter on the inlet cold water pipe measures the flow rate. All the valves are pneumatic. The pressure signals for the level and the flow rate are converted to appropriate current signals (4-20 mA). Temperature signals obtained through the thermocouples are also connected to current signals (4-20 mA). The process was configured as a 2×2 MIMO system. Water level in the tank and the water temperature were selected as the two controlled variables and the valve openings (manipulating cold water flow and steam flow) were selected as the two manipulated variables. The first thermocouple at the exit of the tank was selected for the measurement of the water temperature. The water level in the tank varies only with the inlet water flow and is invariant to changes in the steam flow rate. Thus the water level and inlet water flow rate act as a SISO system. On the other hand, the water temperature varies with both steam flow rate and the inlet water flow rate i.e. a MISO system is developed with these three variables.

An ARMarkov model was developed for the above process using real time, experimental open-loop input-output data and the ARMarkov identification method. The ARMarkov model order was $n = 2$ and the number of Markov parameters in the model was assumed to be $\mu = 26$. The inputs and outputs were sampled every four

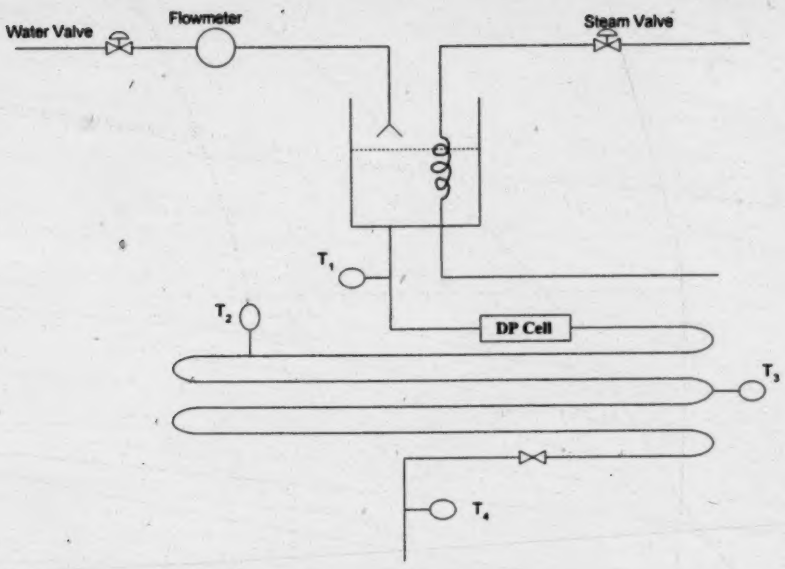


Figure 2.3: Schematic diagram of a pilot scale CSTH.

seconds. The input output data (after removing any 'trend') are plotted in Figure 2.4. The first 1500 data points (only part of them are shown for clarity) were used for the model identification and the remaining 1500 data points (part of them shown) were used for model validation. Estimated outputs are compared with actual outputs in Figure 2.5. Estimated outputs are very close to the actual outputs. This model is used for MPC design later in Chapters 6 and 9.

2.8 Conclusions

- The ARMarkov/LS identification method is extended to MIMO systems.
- The estimated μ Markov parameters are closer to the actual process impulse response than the first μ coefficients of the impulse response obtained by the other linear regression methods such as ARX, FIR.
- The effect of disturbances on the ARMarkov/LS parameter estimates appears (by simulation) to be smaller than on the parameters estimated by the other linear regression methods such as ARX, FIR.

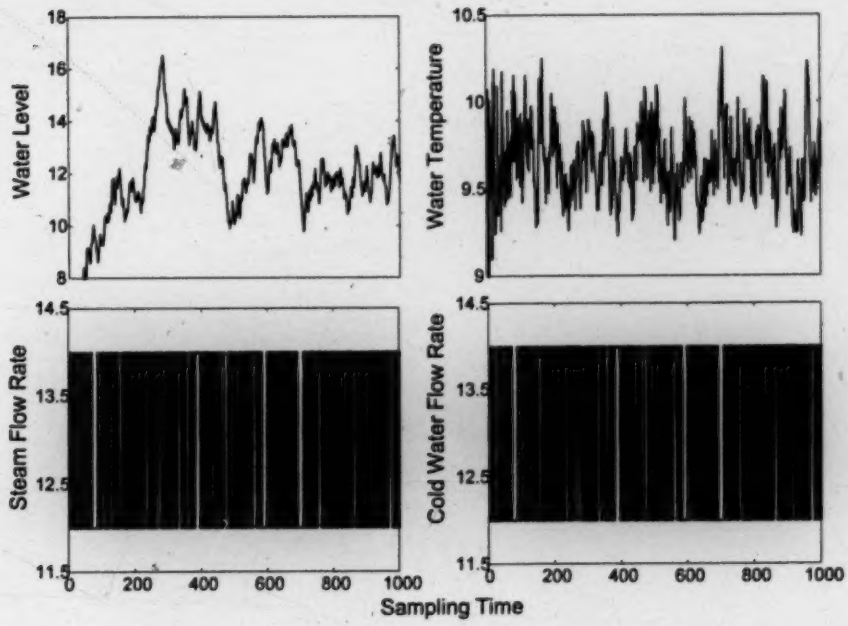


Figure 2.4: Experimental input-output data from a pilot scale CSTH.

- The low order (parametric) model included in the ARMarkov model for the slow dynamics gives satisfactory performance in terms of parameter estimates and/or process response validation compared to other parametric models such as ARX.

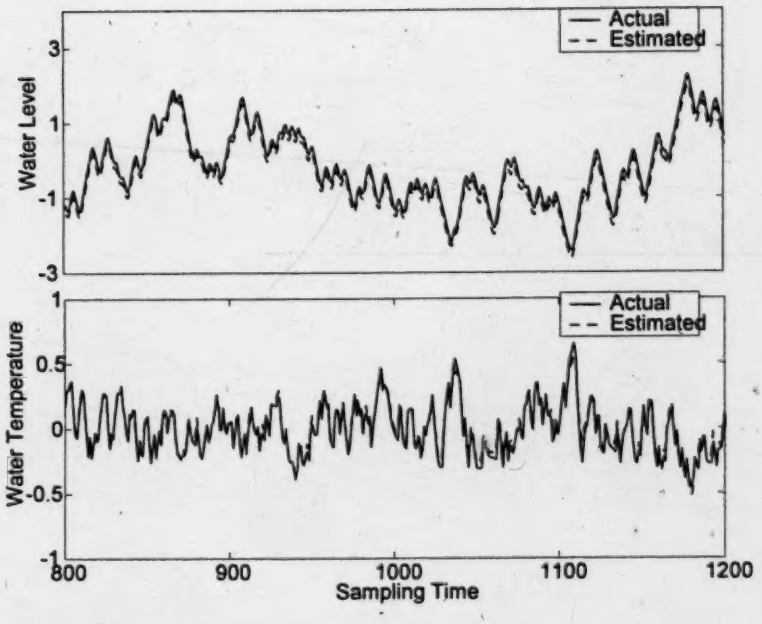


Figure 2.5: ARMarkov model validation for a pilot scale CSTH.

Chapter 3

Estimation of Parametric Residual Model Using AUDI

3.1 Introduction

Linear regression along with Least-squares (LS) method is one of the simplest methods of system identification. Using LS technique, the model parameters are estimated using plant input-output data. The model order is selected by the user and in practical cases, the user has to run several LS trials to select the model order when the actual plant order is unknown. Therefore the computational cost and time increase as the number of trials increases.

To avoid having to complete several LS trials to select the model order, Niu *et al.*(1990, 1992)proposed an Augmented Upper Diagonal Identification (AUDI) method to estimate the model order from 1 to n where n is the maximum order. Advantages of using the AUDI method include: the model order, loss function corresponding to each model order and model parameters corresponding to each model order are simultaneously available at the cost of one trial LS method. Therefore, it is a matter of only selecting the model order corresponding to the acceptable loss function. In the AUDI method, the regressor vector is re-ordered followed by the decomposition of the covariance matrix using LU and UD factorization methods (Bierman 1977, Dahlquist and Bjorck 1974). Niu *et al.* (1992) showed that the AUDI-ARMAX method has numerical stability provided by the LU, DU factorization. The AUDI method, as summarized by Niu *et al.* (1992),

(a) simultaneously estimates the parameters and loss functions for all model orders

from 1 to a user-specified upper limit n with approximately the same computational effort as nth order Recursive least-squares (RLS)

(b) is inherently a least-squares algorithm

(c) has good numerical properties and stability

(d) provides a basis for adding features such as on/off mechanism in the identification algorithm

(e) is much easier to interpret and understand than the original UD algorithm.

Banerjee and Shah (1996) formulated the AUDI method for orthonormal functions to simultaneously estimate multiple model orders and the corresponding loss functions. They applied the AUDI method to Laguerre, Kautz, FIR and Markov-Laguerre models. They also pointed out the effect of noisy data and how it can be handled by using LU decomposition instead of UDU^T decomposition.

As discussed in Chapter 2, the ARMarkov model has two parts: non-parametric and parametric. The order of the non-parametric part is user specified depending on the application such as time delay/interactor matrix estimation and/or predictive controller design and so on. The order of the parametric residual part of the AR-Markov model can be approximated by a low order model. However, proper selection of the parametric model order would lead to more accurate parameter estimation and more reliable MPC design where the ARMarkov model is used to design ARM-MPC in Chapter 8. It is recommended in this chapter that the AUDI method be used to estimate the parametric residual model order in the ARMarkov model selection.

In Section 3.2, the AUDI method proposed by Niu *et al.*and (1990, 1992)is briefly reviewed. In Section 3.3, the regressor vector corresponding to the ARMarkov model is re-arranged. The AUDI formulation which includes the estimation of the model parameters and the loss functions of the ARMarkov method are described in Section 3.4. Examples of the ARMarkov-AUDI algorithm are given in Section 3.5.

3.2 Introduction to the AUDI Method

A linear, time invariant, discrete time model is written as

$$y(k) = \phi^T(k)\,\theta + \varepsilon(k) \tag{3.1}$$

where $y(k)$ is the process output, θ is the model parameter vector, ε is white noise and ϕ is called the regressor vector or data vector consisting of present/past available input-output data such that

$$\theta = [a_1, a_2, \cdots, a_n, b_1, b_2, \cdots b_n]^T \tag{3.2}$$

$$\phi(k) = [-y(k-1), -y(k-2) \cdots - y(k-n), u(k-1), u(k-2) \cdots u(k-n)]^T \tag{3.3}$$

where n is the model order.

3.2.1 Regressor Vector and Information Matrix for the AUDI Method

The basis for the AUDI formulation is the re-ordered data/regressor vector and the UD factorization of the information accumulation matrix (IAM). The regressor vector in (3.3) is rearranged as

$$\phi_n(k) = [-y(k-n), u(k-n) \cdots - y(k-1), u(k-1)]^T$$

and then an augmented regressor vector is defined by Niu *et al.* (1990) as

$$\begin{aligned} \phi_{na}(k) &= [-y(k-n), u(k-n) \cdots - y(k-1), u(k-1), -y(k)]^T \\ &= [\phi_n^T(k), -y(k)]^T \end{aligned} \tag{3.4}$$

The arrangement of the elements in the augmented regressor vector is different (input-output at each sampling time is paired) from the arrangement in the traditional LS regressor vector in (3.3) and the current output is included. The parameter vector, θ is rearranged as

$$\theta_n = [a_n, b_n \cdots a_1, b_1, -1]^T \tag{3.5}$$

A covariance matrix, called the data product moment matrix (DPMM), is defined by Niu *et al.* (1990) and Niu (1994) as

$$S_n(k) = \left[\sum_{i=1}^{k} \phi_{na}(i,n)\, \phi_{na}^T(i,n)\right]_{(2n+1)\times(2n+1)} \tag{3.6}$$

The information accumulation matrix (IAM) defined by Niu *et al.* (1990) or augmented information matrix (AIM) by Niu (1994), Niu *et al.* (1990) is the inverse of the DPMM and expressed as

$$C_n(k) = \left[\sum_{i=1}^{k} \phi_{na}(i,n)\,\phi_{na}^T(i,n)\right]^{-1}_{(2n+1)\times(2n+1)} \tag{3.7}$$

The IAM is decomposed using UDU^T factorization (Bierman 1977, Ljung and Soderstrom 1983) to get the model parameters and loss functions corresponding to the model orders 1 to n.

3.2.2 Decomposition of IAM or DPMM and Parameter Estimation in the AUDI Algorithm

The UDU^T decomposition of IAM and the LDL^T decomposition of DPMM are given by Niu *et al.*(1990, 1992)as

$$C_n(k) = U_n(k)\,D_n(k)\,U_n(k) \tag{3.8}$$

$$S_n(k) = L(k)\,\Xi(k)\,L(k)^T \tag{3.9}$$

$$\begin{aligned} C_n(k) &= S_n(k)^{-1} \\ \Xi(k) &= D_n(k)^{-1} \\ U_n(k) &= L_n^{-T}(k) \end{aligned}$$

where U_n is a $(2n+1)\times(2n+1)$ upper triangular matrix with 1's in the principal diagonal and is expressed as

$$U_n(k) = \begin{bmatrix} 1 & \widehat{\alpha}_0(k-n) & & & & & & \\ & 1 & \widehat{\theta}_1(k-n) & & & & & \\ & & 1 & & & & & \\ & & & \ddots & \widehat{\theta}_{n-1}(k-1) & \widehat{\alpha}_{n-1}(k-1) & \widehat{\theta}_n(k) & \\ & & 0 & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \end{bmatrix} \tag{3.10}$$

D_n is a $(2n+1) \times (2n+1)$ diagonal matrix of the form

$$D_n(k) = diag\{J_0^{-1}(k-n), L_0^{-1}(k-n), \cdots, J_{n-1}^{-1}(k-1), L_{n-1}^{-1}(k-1), J_n^{-1}(k)\} \tag{3.11}$$

and $\Xi_n(k)$, also a $(2n+1) \times (2n+1)$ matrix, includes the loss functions in the form

$$\Xi_n(k) = diag\{J_0(k-n), L_0(k-n), \cdots, J_{n-1}(k-1), L_{n-1}(k-1), J_n(k)\}$$

where $\widehat{\theta}_{n-i}(k-i)$ and $J_{n-i}(k-i)$ are the parameter vectors and loss functions respectively for the $(n-i)th$ order model, $i \in [0, n-1]$. $\widehat{\alpha}_{n-i}(k-i)$ and $L_{n-i}(k-i)$ are intermediate variables in $U_n(k)$ and $D_n(k)$. For example, $\widehat{\theta}_n(k)$ in the last column of the $U_n(k)$ matrix gives the parameter vector in (3.5) for the *nth* order model in (3.1) and the last scalar term in the $D_n(k)$ matrix provides the inverse of the loss function corresponding to the *nth* order model such that

$$\begin{aligned} \widehat{\theta}_n(k) &= [a_n, b_n \cdots a_1, b_1]^T \\ J_n(k) &= \sum_{i=1}^{k} (y(k) - \widehat{y}(k))^2 \end{aligned}$$

The advantage of the AUDI algorithm over LS method is that the $U_n(k)$ and $D_n(k)$ matrices contain the parameter estimates and the loss functions corresponding to the model orders from 1 to n during the same step of identification. Moreover, the UD factorization is numerically more stable than LS methods. The properties of $U_n(k)$ and $D_n(k)$ matrices are discussed in detail in (Niu *et al.* 1990, Niu *et al.* 1992, Niu 1994).

Detailed derivations of the LDL^T and UDU^T factorization for the DPMM and IAM are given in Appendix A.1. The algorithm for LU and/or UDU^T decomposition and the inverse of lower/upper triangular matrix can be found in (Bierman 1977, Dahlquist and Bjorck 1974, Niu 1994). The following remarks apply to the decomposition of the DPMM and IAM to obtain the model parameters and loss functions using the AUDI formulation (Niu 1994).

Remark 3.1 *It is sufficient to form the IAM or the DPMM to obtain the model parameters and the loss functions corresponding to the models of order* 1 *to* n *since the IAM or the DPMM contains all the necessary information.*

Remark 3.2 *From the UDU^T factorization of the IAM, the $U(k)$ matrix gives the parameter estimates and the $D(k)^{-1}$ matrix gives the loss functions. From the LDL^T factorization of the DPMM, the $L(k)^{-1}$ matrix gives the parameter estimates and the $\Xi(k)$ matrix gives the loss functions.*

Remark 3.3 *For off-line batch processing, the recommended procedure is to decompose the DPMM since it is easier to construct, involves the inversion of a lower triangular matrix with unity diagonal elements and the inverse is always available. It does not involve the inversion of a full matrix.*

Remark 3.4 *For on-line process identification or for control, the decomposition of the IAM is recommended because it takes the advantage of Bierman's UD factorization algorithm (Bierman 1977) in the recursive AUDI algorithms and the UDU^T decomposition directly gives the parameter matrix, $U(k)$ and requires the inversion of scalars to obtain the loss function.*

3.2.3 Simulation Example

Example 3.1

Consider the second order process

$$G(z) = \frac{z + 0.5}{z^2 - 1.5z + 0.7} \tag{3.12}$$

AUDI was formulated for this process following the procedure described in Section 3.2 using the highest model order 4. The loss functions for the model orders 1 through 4 are plotted in Figure 3.1. It is clear that model order 2 can be used reasonably for this process because for model order > 2, the loss function i.e. the performance cost does not decrease significantly.

Parameters corresponding to model order 2 is given by the $(2 * 2 + 1) = 5th$ column of the U matrix and shown in Table 3.1 (the highlighted column). The estimated parameters are very close to the actual parameters (compared in Table 3.2) that shows the reliability of the AUDI method.

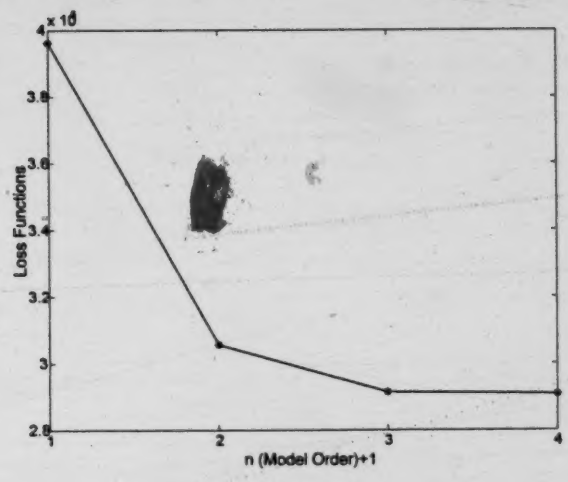


Figure 3.1: Loss functions for different model orders in the AUDI formulation.

Table 3.1: Parametres corresponding to different models in the AUDI formulation

model order	0	1	2	3
Parameter vector	0.1985	1.7049	0.4993	−0.5001
	−0.8881	−0.8881	0.7001	−0.7001
	1.0000	−0.4256	1.0000	−0.5001
	0.0000	1.0000	−1.5002	2.2002
	0.0000	0.0000	−0.0006	1.0000
	0.0000	0.0000	1.0000	−2.5001
	0.0000	0.0000	0.0000	0.0000
	0.0000	0.0000	−0.0000	1.0000
	0.0000	0.0000	−0.0000	−0.0000

3.3 Re-arrangement of the Regressor Vector for the ARMarkov Model

The key step in the AUDI formulation is the formation of the DPMM or the IAM that is constructed using the regressor vector.

The input-output ARMarkov model in equation (2.18) is rewritten as

$$y(k) = -\sum_{j=1}^{n} \alpha_j y(k-\mu-j+1) + \sum_{j=1}^{\mu} h_{j-1} u(k-j+1) + \sum_{j=1}^{n} \beta_j u(k-\mu-j+1) + \epsilon(k) \quad (3.13)$$

for $k \geq 0$

Table 3.2: Parametres corresponding to different models in the AUDI formulation

Actual Parameter vector	0.5000	0.7000	1.0000	−1.5000	−0.0000	1.0000
Estimated Parameter vector	0.4993	0.7001	1.0000	−1.5002	−0.0006	1.0000

In the AUDI formulation of the ARMarkov model, the number of non-parametric coefficients (i.e. the Markov parameters) is kept fixed and an upper limit, n (e.g. 4 or 5) for the parametric model order is specified.

The input-output data are arranged to form the regressor vector corresponding to the nth order residual parametric model as

$$\begin{aligned}\phi_{nara}(k) &= [-y(k), u(k), \cdots, u(k-\mu+1), u(k-\mu), -y(k-\mu), \\ &\quad \cdots, u(k-\mu-n+1), -y(k-\mu-n+1)]^T \qquad (3.14)\\ &= \left[-y(k), \phi_{nar}^T(k)\right]^T \qquad (3.15)\end{aligned}$$

Note that the current output data is placed at the beginning of the regressor vector instead of at the last position in the original AUDI formulation by Niu (1994). The arrangement of the first μ elements in (3.14) is similar to the AUDI formulation by Banerjee and Shah (1996)for orthonormal functions e.g. impulse response functions and the last $2n$ elements are arranged in the similar manner as was done in the original AUDI formulation by Niu (1994)except that the elements are placed here in a descending order instead of an ascending order. Therefore, the AUDI-ARMarkov model structure is a combination of the AUDI-ARMAX formulated by Niu (1994)and the AUDI-impulse response formulated by Banerjee and Shah (1996).

The parameter vector corresponding to the nth order parametric model and the regressor vector $\phi_{nar}(k)$ in (3.14) is

$$\theta = \left[\, h_0 \;\; h_1 \;\; \cdots \;\; h_{\mu-1} \;\; \beta_1 \;\; \alpha_1 \;\; \cdots \;\; \beta_{na} \;\; \alpha_{na} \,\right]^T \qquad (3.16)$$

However, the parameter vector corresponding to the regressor vector $\phi_{nara}(k)$ in (3.14) is

$$\theta_a = \left[\, \zeta_1 \;\; \zeta_2 \;\; \cdots \;\; \zeta_{\mu+2n} \;\; \zeta_{\mu+2n+1} \,\right]^T \qquad (3.17)$$

3.4 AUDI Formulation of the ARMarkov Model

The data product moment matrix (DPMM) and the information accumulation matrix (IAM) corresponding to the regressor vector in (3.14) can be written as

$$S_{n(arm)}(k) = \left[\sum_{i=1}^{k} \phi_{nara}(i,n)\,\phi_{nara}^{T}(i,n)\right]_{(\mu+2n+1)\times(\mu+2n+1)} \tag{3.18}$$

$$C_{n(arm)}(k) = S_n(k)^{-1} = \left[\sum_{i=1}^{k} \phi_{nara}(i,n)\,\phi_{nara}^{T}(i,n)\right]^{-1}_{(\mu+2n+1)\times(\mu+2n+1)} \tag{3.19}$$

The LDL^T factorization of the DPMM and/or the UDU^T factorization of the IAM lead to the solution in terms of the parameter vector and loss functions of the AUDI-ARMarkov problem.

3.4.1 Solution of the Parameter Vector in the AUDI-ARMarkov method

The last column of the U_{an} matrix (obtained from the UDU^T decomposition of $C_{n(arm)}(k)$) includes the solution of the parameter vector θ_a in (3.17) as

$$U_{an}(k) = \begin{bmatrix} 1 & \widehat{\alpha}_{a0}(k-n) & & & & & & \\ & 1 & \widehat{\theta}_{a1}(k-n) & & & & & \\ & & 1 & & & & & \\ & & & \ddots & \widehat{\theta}_{an-1}(k-1) & \widehat{\alpha}_{an-1}(k-1) & \widehat{\theta}_{an}(k) \\ & & 0 & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix} \tag{3.20}$$

This parameter vector cannot be directly used as the identified ARMarkov model. The solution of the ARMarkov parameter vector $\widehat{\theta}_n(k)$ is obtained by the modification of $\widehat{\theta}_{an}(k)$ as

$$[1 \;\; \widehat{\theta}_n^T]^T = \frac{1}{\zeta_1}\left[\widehat{\theta}_{an}^T(k)\right]^T \tag{3.21}$$

3.4.2 Loss Functions with Multiple Model Order

The diagonal matrix D_n is obtained from the UDU^T decomposition of $C_{n(arm)}(k)$ as

$$\begin{aligned} D_n(k) = \; & diag\{J_0^{-1}(k,0), \cdots, J_{\mu-1}^{-1}(k,\mu-1), J_{\mu}^{-1}(k-1), L_0^{-1}(k), \cdots, \\ & J_{\mu+n-1}^{-1}(k-1), L_{n-1}^{-1}(k)\} \end{aligned} \tag{3.22}$$

and $\Xi_n(k)$ includes the loss functions in the form

$$\Xi_n(k) = diag\{J_0(k,0), \cdots, J_{\mu-1}(k,\mu-1), J_\mu(k-1), L_0(k), \cdots, J_{\mu+n-1}(k-1), L_{n-1}(k)\}$$

where $\Xi_n(k)$ is obtained from the UDU^T decomposition of $S_{n(arm)}(k)$. $J_i(k,i)$ is the loss function for the *ith* order model, $i \in [0, \mu+n-1]$ and $L_i(k,i)$ are intermediate variables in $D_n(k)$.

3.5 Simulation Example

Example 3.2

Consider the third order SISO system in (2.38) of Chapter 2 with the transfer function,

$$G(z) = \frac{0.0077z^{-1} + 0.0212z^{-2} + 0.0036z^{-3}}{1 - 1.9031z^{-1} + 1.1514z^{-2} - 0.2158z^{-3}} \tag{3.23}$$

The AUDI was formulated for this process following the procedure described in Section 3.3. The number of Markov parameters was kept constant at $\mu = 10$ and the highest model order for the parametric residual model was $na = 4$. The loss functions for the model orders $\mu+1$ through $\mu+4$ are plotted in Figure 3.2. It is clear that parametric model order 2 or 3 could be used without significant additional performance cost, but the actual process order is 3. Therefore, a lower order (i.e. 2) can be used without much loss in performance.

The first μ actual and estimated Markov parameters are plotted in Figure 3.3. The estimated parameters are very close to the actual parameters.

3.6 Conclusions

- The AUDI method is reformulated for use in estimating an ARMarkov model.
- Estimation of model order is done on the basis of loss functions for different model orders and compared to the actual model order for known processes. From simulation, the estimated model orders appear to be reliable.

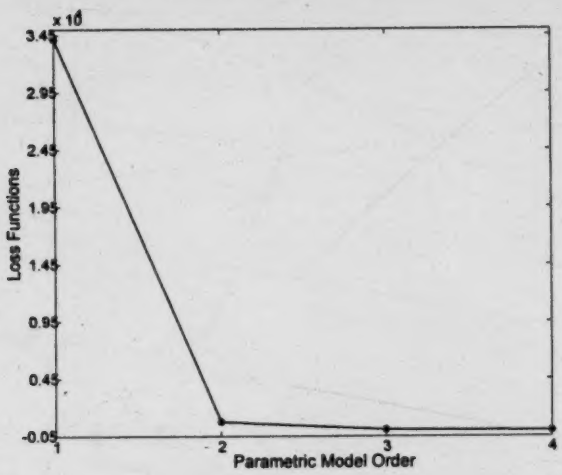


Figure 3.2: Loss functions for different parametric model orders in the AUDI formulation of the ARMarkov model.

- Simulation results show that for a specified number of Markov parameters, the Markov parameters can be identified accurately using a parametric model order smaller than the actual parametric model order.

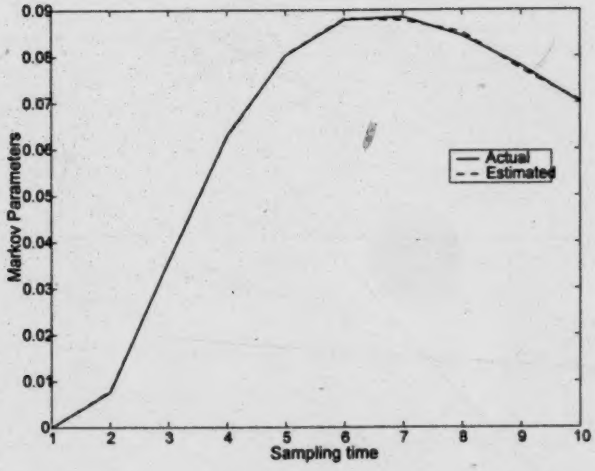


Figure 3.3: Actual and estimated Markov parameters using the AUDI formualtion of the ARMarkov model.

Chapter 4

Statistical Analysis of the ARMarkov Parameter Estimates[1]

4.1 Introduction

When a model is identified using process input-output observations, a very common question is "How good is the model ?". When answering this question, a second question comes to any mind: "How to define a 'good' model ?". One common practice in modeling or system identification is to assume a model structure and estimate a parameter vector, $\hat{\theta}$ for that model by minimizing some errors in predicting present/future output(s) based on information available from the process. Then the 'goodness' of the model is characterized by the 'goodness' of the parameter estimates. If the actual process parameters are known, the estimated parameters can be compared to the actual parameters and closeness of these two sets of parameters certifies the model as 'good'. But in most practical cases, the actual process parameters are unknown to the user and the model is a 'black box' type i.e. no structure and/or parameters of the process are known. In these cases, statistical properties of the estimated parameters are analyzed to assess the 'good-ness' of the parameters.

In the following discussion of statistical analysis, only the relevant properties in the present context of parameter estimation are discussed rather than bringing the whole statistical literature into discussion. The statistical properties of the model parameters commonly analyzed in system identification are: bias, consistency, variance/covariance and confidence intervals. Bias of the model parameters is defined as:

[1]Contents of this chapter was presented at the 49th CSCHE conference, Saskatoon, Canada, Oct. 1999 and is a part of the paper published in the Chemical Engineering Science in May 2000.

when the model set does not contain the actual process, the estimated parameters are called *biased* estimates and vice versa. Since the ARMarkov model described in Chapter-2 is overparameterized (Akers and Bernstein 1997), the bias of the parameters is not discussed here. The other properties (e.g. consistency, variance/covariance and confidence intervals) of the ARMarkov parameters are discussed in this chapter.

ARX and FIR are the simplest of the most widely used identification methods which use linear regression methods. The ARMarkov method also uses linear regression. Since the ARX and FIR methods are parametric and nonparametric respectively and the ARMarkov method is a blend of these two methods, it is intuitively expected that the ARMarkov method will have the properties which are in between the properties of the parametric and non-parametric formulations.

The *consistency* of the ARMarkov parameters is analyzed and compared with the consistency of the parameters estimated by ARX and FIR methods in Section 4.2. The *variance/covariance* properties are discussed in Section 4.3 and the *confidence intervals* on the estimated parameters are analyzed in Section 4.4. Illustrative examples on covariance analysis and confidence intervals are shown in Section 4.5.

4.2 Consistency of the Least-squares Estimates

The *consistency* of the estimated parameters is defined as: 'the estimated parameters, $\hat{\theta}$ are consistent if $\hat{\theta}$ converges to the actual parameters θ_0 as the number of data points $N \to \infty$' (Ljung 1987).

The general structure of the actual process is assumed to be

$$y = Gu + He \tag{4.1}$$

$$= \frac{B}{A}u + He \tag{4.2}$$

and the model is written as

$$\hat{y} = \hat{G}u + \hat{H}e \tag{4.3}$$

where H is noise structure, A, B are polynomials in z, $\hat{G}$ and $\hat{H}$ are different for different model structures and e is white noise with zero mean and covariance $\lambda^2 I$ where λ represents the variance of e.

In input-output form, the process can be expressed as

$$y(k) = \phi^T(k)\theta_0 + v_0(k) \tag{4.4}$$

where ϕ is the regressor vector, θ_0 represents the actual process parameter vector and v_0 is the process noise with $v_0 = AHe$.

The structures of the linear regression models are as follows:

$$\begin{aligned} ARX \;:\; \widehat{y}(k) &= -a_1 y(k-1) - \cdots - a_n y(k-n) + b_0 u(k) + b_1 u(k-1) + \cdots \\ &\quad + b_n u(k-n) \end{aligned} \tag{4.5}$$

$$FIR \;:\; \widehat{y}(k) = f_0 u(k) + f_1 u(k-1) + \cdots + f_{M-1} u(k-M+1) \tag{4.6}$$

$$\begin{aligned} ARMarkov \;:\; \widehat{y}(k) &= -\alpha_1 y(k-\mu) - \cdots - \alpha_n y(k-\mu-n+1) \\ &\quad + h_0 u(k) + h_1 u(k-1) + \cdots + h_{\mu-1} u(k-\mu+1) \\ &\quad + \beta_1 u(k-\mu) + \cdots + \beta_n u(k-\mu-n+1) \end{aligned} \tag{4.7}$$

where n is the order of the ARX model, M represents the number of FIR coefficients and μ is the number of Markov parameters in ARMarkov model. The above three models can also be written in terms of regressor vectors and model parameters as

$$ARX \;:\; \widehat{y}(k) = \phi_{ARX}^T(k)\theta_{ARX} + v_{ARX}(k) \tag{a}$$

$$FIR \;:\; \widehat{y}(k) = \phi_{FIR}^T(k)\theta_{FIR} + v_{FIR}(k) \tag{b}$$

$$ARMarkov \;:\; \widehat{y}(k) = \phi_{ARM}^T(k)\theta_{ARM} + v_{ARM}(k) \tag{c}$$

Now assume that *the input signal is persistently exciting and the model contains the actual process (no bias).* Then according to Ljung (1987), the Least-Squares Estimate (LSE) is consistent i.e. $\widehat{\theta}$ converges to θ_0 as the number of data points, $N \rightarrow \infty$ if

1. v_0 is white noise (not the case in most practical applications) or
2. the input sequence is independent of the noise sequence and there is no output term in the regression vector.

In the FIR model, there is no output term in the regressor vector, so the estimated parameters converge to the actual ones as $N \rightarrow \infty$, but in the ARX method, the parameter estimate is not consistent in the presence of non-white noise (Ljung 1987, Soderstrom and Stoica 1989) because of the output terms in the regressor.

Lemma 4.1 *The estimated Markov parameters in the ARMarkov model are consistent, although the residual parameters equivalent to the ARX parameters in the ARMarkov model are not.*

Proof. *For the ARMarkov method, the least-squares estimate can be written as*

$$\widehat{\theta}^{LS} = \theta_0 + \left[\frac{1}{N}\sum_{i=1}^{N}\phi(k)\phi^T(k)\right]^{-1}\frac{1}{N}\sum_{i=1}^{N}\phi(k)\nu_0(k) \tag{4.8}$$

$$= \theta_0 + (E(\phi(k)\phi^T(k)))^{-1}E(\phi(k)\nu_0(k)) \quad as\ N \to \infty \tag{4.9}$$

Let

$$\theta_0 = \begin{bmatrix} \theta_{Mark} \\ \theta_{AR} \end{bmatrix},\ \phi = \begin{bmatrix} \phi_{Mark} \\ \phi_{AR} \end{bmatrix} \tag{4.10}$$

$$\phi(k)\nu_0(k) = \begin{bmatrix} \phi_{Mark}\nu_0(k) \\ \phi_{AR}\nu_0(k) \end{bmatrix} \tag{4.11}$$

where ϕ_{Mark} corresponds to the present and past input terms. Hence $E(\phi_{Mark}(k)\nu_0(k)) = 0$,. but $E(\phi_{AR}(k)\nu_0(k)) \neq 0$ since ϕ_{AR} involves some output terms. Therefore, as $N \to \infty$

$$\widehat{\theta}^{LS} = \begin{bmatrix} \theta_{Mark} \\ \theta_{AR} \end{bmatrix} + \begin{bmatrix} 0 \\ (E(\phi(k)\phi^T(k)))^{-1}E(\phi(k)\nu_0(k)) \end{bmatrix} \tag{4.12}$$

$$= \begin{bmatrix} \theta_{Mark} \\ \theta_{AR} + (E(\phi(k)\phi^T(k)))^{-1}E(\phi(k)\nu_0(k)) \end{bmatrix} \tag{4.13}$$

$$\widehat{\theta}^{LS}_{Mark} = \theta_{Mark} \tag{4.14}$$

■

Example 4.1

Lemma 4.1 can be illustrated by considering the following first order process (Soderstrom and Stoica 1989) with non-white disturbances

$$y(k) = -a_1 y(k-1) + b_0 u(k-1) + c_0 e(k) + c_1 e(k-1) \tag{4.15}$$

Assume an ARMarkov model with two Markov parameters h_0 and h_1 so that

$$\widehat{y}(k) = -\alpha_1 y(k-2) + h_0 u(k) + h_1 u(k-1) + \beta_0 u(k-2) \tag{4.16}$$

Here u is an independent input sequence with variance σ^2 and e is white noise with variance λ^2.

As $N \rightarrow \infty$, the least-squares parameter estimates are

$$h_0 = 0 \tag{4.17}$$

$$\begin{bmatrix} h_1 \\ \alpha_1 \\ \beta_0 \end{bmatrix} = [R^*]^{-1} \begin{bmatrix} Ey(k)u(k-1) \\ Ey(k)y(k-2) \\ Ey(k)u(k-2) \end{bmatrix} \tag{4.18}$$

$$R^* = \begin{bmatrix} Eu(k-1)^2 & -Ey(k-2)u(k-1) & Eu(k-1)u(k-2) \\ -Ey(k-2)u(k-1) & Ey(k-2)^2 & -Ey(k-2)u(k-2) \\ Eu(k-1)u(k-2) & -Ey(k-2)u(k-2) & Eu(k-2)^2 \end{bmatrix} \tag{4.19}$$

$$R^* = \begin{bmatrix} \sigma^2 & 0 & 0 \\ 0 & Ey(k-2)^2 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix} \tag{4.20}$$

$$Ey(k-2)^2 = \frac{b_0^2\sigma^2 + (1 + c_0^2 - a_1 2c_0)\lambda^2}{1 - a_1^2}, \quad Ey(k)u(k-1) = b_0\sigma^2 \tag{4.21}$$

$$Ey(k)u(k-2) = -a_1 b_0 \sigma^2, \tag{4.22}$$

$$Ey(k)y(k-2) = a_1^2 \frac{b_0^2\sigma^2 + (1 + c_0^2 - a_1 2c_0)\lambda^2}{1 - a_1^2} + c_0\lambda^2 \tag{4.23}$$

Therefore,

$$\begin{bmatrix} h_1 \\ \beta_0 \end{bmatrix} = \begin{bmatrix} b_0 \\ -a_1 b_0 \end{bmatrix} \tag{4.24}$$

Remark 4.1 *The Markov parameters, h_0 and h_1, are consistent (transformed into actual parameters (Akers and Bernstein 1997), also see Chapter-2); whereas, α_1, which is a function of the noise/disturbance parameters never converges to the actual parameter. Since the ARMarkov model is a blend of the FIR and ARX models (equations (4.5)-(4.7)), it is consistent in the parameters corresponding to the FIR models but inconsistent in the residual parameters corresponding to the ARX model. Note that in the estimation of the interactor matrix (described in Chapter 5) used in performance assessment and in the dynamic matrix construction in MPC applications only the consistent Markov parameters are needed.*

4.2.1 Consistency Under Closed-loop Condition

When closed-loop data are used for the ARMarkov identification or for on-line updating of the ARMarkov model parameters e.g. during on-line recursive identification as discussed in Chapter 2, the input sequence is no longer independent of the noise sequence. In that case, the consistency of the estimated Markov parameters is discussed in this subsection.

Rewrite the ARMarkov model as

$$\begin{aligned} ARMarkov \;:\; y(k) &= h_0 u(k) + h_1 u(k-1) + \cdots + h_{\mu-1} u(k-\mu+1) \\ &+\beta_1 u(k-\mu) + \cdots + \beta_n u(k-\mu-n+1) \\ &-\alpha_1 y(k-\mu) - \cdots - \alpha_n y(k-\mu-n+1) + x_d(k) \end{aligned}$$

where $x_d(k)$ is the disturbance sequence.

Assume that the noise sequence is white and the time delay of the process is d. The current and $(d-1)$ future values of the output, y depend only on past values of the process input which are NOT a function of the current and $(d-1)$ future noise values. Therefore, the estimates of the first d Markov parameters under closed-loop condition are consistent.

If the time delay of the process is d, then the first d Markov parameters should be statistically insignificant. Hence the estimate of the time delay of the process is consistent using closed-loop data and ARMarkov identification.

4.3 Covariance of the Estimated Parameters

4.3.1 Covariance Estimates for Systems with White Noise

The output prediction error, $\varepsilon(k)$ in (2.25) and the estimated parameter vector in (2.28) can be rewritten as

$$\varepsilon(k) = y(k) - \hat{y}(k) \tag{4.25}$$

$$\hat{\mathbf{W}}_\mu = \left[\frac{1}{N}\sum_{k=1}^{N} y(k)\boldsymbol{\phi}_\mu^T(k)\right]\left[\frac{1}{N}\sum_{k=1}^{N} \boldsymbol{\phi}_\mu(k)\boldsymbol{\phi}_\mu^T(k)\right]^{-1} \tag{4.26}$$

Assume that, for SISO systems, the error terms, ε in (4.25) are statistically independent having 'zero' mean and variance λ^2 and the estimates $\widehat{\theta}^{LS}$ denoted by $\hat{\mathbf{W}}_\mu$ in (4.26) are the unbiased estimates of the actual parameters $\mathbf{W}_\mu$ i.e. $E(\hat{\mathbf{W}}_\mu) = \mathbf{W}_\mu$. See (Johnson and Wichern 1988) for the proof. The *covariance matrix* of the estimates is

$$cov(\hat{\mathbf{W}}_\mu) = \lambda^2 \left[\frac{1}{N} \sum_{i=1}^{N} \phi(k)\phi^T(k) \right]^{-1} \tag{4.27}$$

The diagonal elements of the *covariance matrix* represent the estimated variances of the estimators. The estimated squared standard error (SSE) of $\hat{\mathbf{W}}_\mu$ can be written as

$$SSE = \sum_{k=1}^{N} \varepsilon^T(k)\varepsilon(k) \tag{4.28}$$

$$= \sum_{k=1}^{N} y(k)^2 - \hat{\mathbf{W}}_\mu^T \left(\frac{1}{N} \sum_{k=1}^{N} y(k)\phi^T(k) \right) \tag{4.29}$$

and the estimate of the error variance is written as

$$\widehat{\lambda}^2 = \frac{SSE}{N-p} = \frac{\sum_{k=1}^{N} y(k)^2 - \hat{\mathbf{W}}_\mu^T \left(\frac{1}{N} \sum_{k=1}^{N} y(k)\phi^T(k) \right)}{N-p} \tag{4.30}$$

where p is the total number of parameters to be estimated and N is the number of data points.

When the actual process is subjected to white noise, the noise estimates determined by the LS methods are independently distributed with zero mean and variance $\widehat{\lambda}^2$. In this case, the linear regression method is equivalent to the Prediction Error Method (PEM). According to Ljung (1987), for PEM, the total variance of the parameter estimates (expressed in terms of the process model variance) is proportional to the ratio of the number of model parameters to the number of data points. i.e.

$$var(\widehat{G}_{ARX}) \propto \frac{2n}{N} \tag{4.31}$$

$$var(\widehat{G}_{ARMark}) \propto \frac{\mu + 2n}{N} \tag{4.32}$$

$$var(\widehat{G}_{FIR}) \propto \frac{M}{N} \tag{4.33}$$

The proportionality constant is the same for all the three models and equals the ratio of the input spectrum to the noise spectrum.

Lemma 4.2 *For the same input-output data set*

$$var(\widehat{G}_{FIR}) > var(\widehat{G}_{ARMark}) > var(\widehat{G}_{ARX}) \tag{4.34}$$

Proof. *Assume $M \geq \mu + 2n$ and $\mu > 1$. Since N is same for the three methods, the proof directly follows from the expressions of variances in equations (4.31) through (4.33).* ■

For MIMO systems, where the error term $\varepsilon(k)$ in (4.25) is a vector, the above analysis is valid for the special case when the covariance matrix of $\varepsilon(k)$ contains insignificant off-diagonal elements. For the general case i.e. when the covariance matrix of $\varepsilon(k)$ contains significant off-diagonal elements, the analysis can be done by following the procedure discussed in the next subsection for non-white disturbances.

4.3.2 Covariance Estimates for Systems with Non-white Noise

Now assume that the actual process is subjected to non-white noise, ν_0 or white noise which contains significant off-diagonal elements in the covariance matrix. The covariance of the noise/disturbances can be written as

$$E\nu_0\,\nu_0^T = R \tag{4.35}$$

where R is a positive definite matrix. Then the covariance of the parameter estimates (Soderstrom and Stoica 1989) is

$$\begin{aligned} cov(\widehat{\theta}) &= \left[\frac{1}{N}\sum_{k=1}^{N}\phi(k)\phi^T(k)\right]^{-1}[\phi(1)\ \cdots\ \phi(N)]\,R\begin{bmatrix}\phi^T(1)\\ \vdots \\ \phi^T(N)\end{bmatrix} \\ &\quad \left[\frac{1}{N}\sum_{i=1}^{N}\phi(k)\phi^T(k)\right]^{-1} \end{aligned} \tag{4.36}$$

which is a non-minimal, positive-definite symmetric matrix. In practice, R is always unknown and replaced by the estimate of the error covariance matrix,

$$\widehat{R} = \frac{1}{N}\sum_{i=1}^{N}\varepsilon(k)\varepsilon(k)^T \tag{4.37}$$

$$= \frac{1}{N}\sum_{i=1}^{N}[y(k) - \widehat{y}(k)]\,[y(k) - \widehat{y}(k)]^T \tag{4.38}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left[y(k) - \phi^T(k)\widehat{\theta}_{LS}\right]\left[y(k) - \phi^T(k)\widehat{\theta}_{LS}\right]^T \tag{4.39}$$

Since the structure of the regressor vector and hence the parameter vector is different in different LS model structures (ARX, FIR, ARMarkov), the methods can not be compared directly. However, since we are specifically interested in the first μ Markov parameters, one approach is to compare the total sum of the variances corresponding to the first μ Markov parameters determined by each LS method. In the parametric methods such as ARX, the Markov parameters are not determined directly. Therefore, the proposed method of comparison is not directly applicable to parametric methods. However, the model parameters can be transformed into the Markov parameters and the variances of the model parameters can be mapped to the variances of the Markov parameters. But this is an indirect measure of variance. Here the sum of the variances of the available model parameters is used as a measure. Assuming that $M > \mu + 2n$, it has been shown *through simulation* (results are shown in section 4.5) that

$$\sum_{i=1}^{\mu} var(f_i^{(FIR)}) > \sum_{i=1}^{\mu} var(h_i^{(ARMark)}) > \sum_{i=1}^{n} \left(var(a_i^{(ARX)}) + var(b_i^{(ARX)}) \right) \quad (4.40)$$

Another basis for comparison of the methods is the largest eigenvalue of the covariance matrices (the largest eigenvalue corresponds to the maxima of the covariance matrix and hence the worst case scenario). It has also been shown *through simulation* (results are shown in section 4.5) that

$$\sigma_{\max}\left(\widehat{R}_{FIR}\right) > \sigma_{\max}\left(\widehat{R}_{ARMark}\right) > \sigma_{\max}\left(\widehat{R}_{ARX}\right) \quad (4.41)$$

where $\sigma_{\max}$ corresponds to the maximum eigenvalue.

Remark 4.2 *The ARMarkov parameter estimates have larger variances than the parametric (ARX) method estimates and smaller variances than the non-parametric (FIR) method estimates for both white and non-white noise disturbances. This is intuitively reasonable since the variances are proportional to the number of estimated parameters.*

4.4 Confidence Intervals on the Parameter Estimates

The confidence intervals on the parameters estimated by the ARMarkov method can be constructed, as in Multiple Linear Regression. Assume that the errors, ϵ in the

output prediction are normally and independently distributed with zero mean and variance λ^2 . Let

$$C = \left[\frac{1}{N}\sum_{i=1}^{N}\phi(k)\phi^T(k)\right]^{-1} \tag{4.42}$$

Then

$$\frac{\hat{\mathbf{W}}_j - W_j}{\sqrt{\hat{\lambda}^2 C_{jj}}} \qquad j = 1, 2, \cdots, p \tag{4.43}$$

has a *t-distribution* with $(N-p)$ degrees of freedom.

The $100(1-\alpha)$ *percent confidence interval on the regression coefficients* W_j, $j = 1, 2, \cdots, p$ is given by

$$\hat{\mathbf{W}}_j - t_{\alpha,N-p}\sqrt{\hat{\lambda}^2 C_{jj}} \leq W_j \leq \hat{\mathbf{W}}_j + t_{\alpha,N-p}\sqrt{\hat{\lambda}^2 C_{jj}} \tag{4.44}$$

where α is the level of significance (Johnson and Wichern 1988). The confidence bounds and hence the confidence intervals involve the parameter variance. As discussed in the previous section, the variance of the first μ Markov parameters determined by the ARMarkov method is smaller than the variance of the first μ Markov parameters determined by the FIR method. Therefore, the μ Markov parameters have tighter confidence bounds than the corresponding FIR parameters. The Markov parameters are not determined directly by the ARX method and hence this analysis is not directly applicable to the ARX method. However, the relationship between confidence bounds and variance implies that the ARX method has the tightest bounds.

Remark 4.3 *As discussed in the above sections, the* ***consistency*** *of the ARMarkov parameter estimates is in between the consistencies of the parameters estimated by the parametric (ARX) and non-parametric (FIR) linear regression methods. The* ***variances*** *and hence the* ***confidence bounds*** *have a similar relationship. The ARX method lacks an analytical measure of consistency and the FIR method has the largest variance due to the large number of parameters. However, the Markov parameters estimated using the ARMarkov method have reasonable estimates of both properties.*

4.5 Simulation Examples

This section describes simulations carried out to analyze the statistical properties of the parameters estimated using the ARMarkov method. Variance/covariance and confidence bounds on the Markov parameters estimated by the ARMarkov method were compared with the properties of the same parameters estimated by other least-squares identification methods such as FIR. As mentioned earlier, due to the indirect measure in the ARX method, the above mentioned properties were not compared by simulation.

Example 4.2

Consider the third order SISO system with the transfer function,

$$G(z) = \frac{0.0077z^{-1} + 0.0212z^{-2} + 0.0036z^{-3}}{1 - 1.9031z^{-1} + 1.1514z^{-2} - 0.2158z^{-3}} \tag{4.45}$$

subjected to colored noise generated by passing white noise through the disturbance transfer function

$$H(z) = \frac{1}{1 - 0.95z^{-1}} \tag{4.46}$$

4.5.1 Variance

The sum of the variances of the Markov parameters as well as the largest eigenvalue of the covariance matrix were determined as described in Section 4.3. Results for white noise disturbances are shown in Table 4.1 and for non-white disturbances in Table 4.2.

Table 4.1: Variance with white noise disturbances

	ARMarkov ($\mu = 10, n = 3$)	FIR ($M = 25$)
total variance	2.51	3.34
σ_{max}	3.16	4.45

As expected, the ARMarkov Markov parameters have lower variance and smaller eigenvalue (largest) than the Markov parameters by FIR.

Table 4.2: Variance with non-white noise disturbances

(Same colored noise described in (4.46))

	ARMarkov ($\mu = 10, n = 3$)	FIR ($M = 25$)
total variance	6.67	12.77
$\sigma_{\max}$	5.31	9.01

4.5.2 Confidence bounds

Consider the process in (4.45) subjected to white noise. The confidence interval of the first μ Markov parameters estimated by the ARMarkov method and by the FIR method were determined by following the procedure described in section 4.4.

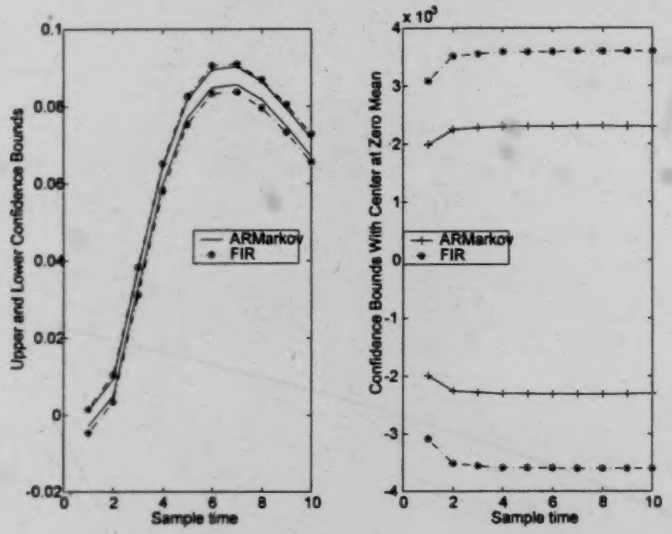


Figure 4.1: Confidence Bounds on Markov Parameters

The results (upper and lower confidence bounds) in Figure 4.1 clearly show that the ARMarkov estimates have a tighter confidence bound than the FIR estimates. Since the Markov parameters are not directly estimated in the ARX method, the confidence bounds for the ARX method were not compared here.

4.6 Conclusions

- The Markov parameters estimated by the ARMarkov/LS method are consistent even in the presence of colored disturbances. The *consistency* of the estimated Markov parameters is better than the consistency of Markov parameters by other parametric linear regression methods such as ARX.
- The estimated μ Markov parameters by the ARMarkov method have lower *variances* and tighter *confidence bounds* than the parameters estimated by other non-parametric methods such as FIR.

Chapter 5

Interactor Matrix Estimation from Markov Parameters[1]

5.1 Introduction

Time delay is a very common characteristic of chemical processes. Even if the process itself does not have a delay, a single delay is introduced when the process/model is discretized. Systems with time delay require special care in designing the controller (Garcia and Morari 1982, Sripada and Fisher 1985, Goodwin and Sin 1984) and it is well known that the achievable control performance is limited by the presence of time delays. In control loop performance assessment, knowledge of time delay is required to generate the basis for performance benchmark (Harris *et al.* 1996, Huang and Shah 1999). If the process time delay can be separated from the delay-free part of the process model, it is easier to design a controller as well as to assess closed-loop performance. In SISO systems, it is easy to estimate and separate the time delay from the delay-free process. In MIMO systems, if all the input-output pairs share the same delay, it would also be easy to factor out the time delay from the process. But in many practical MIMO systems, each input-output pair contains a different time delay. In these cases, it is difficult to separate the time delay part from the delay-free part of the process directly.

Time delays in MIMO systems are defined by the interactor matrix. Wolowich and Falb (1976)proved the existence of the interactor matrix and later it was introduced into the design procedure of feedback controllers (Wolowich and Falb 1976, Tsili-

[1]Contents of this chapter was presented at the 1998 CSCHE conference in London, Ontario, Canada and was also published in the *Chemical Engineering Science*, May 2000.

giannis and Svoronos 1988). Different forms of the interactor matrix and several estimation procedures have been proposed by researchers. The estimation methods used by Wolowich and Falb (1976), Goodwin and Sin (1984)need a priori information on the process transfer functions. Shah *et al.* (1987), Huang and Shah (1999)described a method of unitary interactor estimation that uses a linear combination of process Markov block (impulse response block) matrices and does not require complete information on the process transfer functions.

As stated in (Huang and Shah 1999), "*computationally, a direct identification of the first few Markov parameters is more desirable than an identification of the full transfer function matrix first, followed by its transfer to Markov parameters. Therefore, the factorization of the interactor matrix from the first few Markov parameters is preferred to the factorization of the interactor matrix from the transfer function matrix*". It, therefore, follows that better estimation of the Markov parameters leads to better estimation of the unitary interactor matrix. The ARMarkov model described in Chapter 4 contains the Markov parameters explicitly. Furthermore, the ARMarkov method gives consistent estimates of the Markov parameters even in the presence of colored disturbances as shown by Kamrunnahar *et al.* (2000) and in Chapters 2 and 4.

In this chapter, Markov parameters obtained using the ARMarkov identification method are used to estimate the unitary interactor matrix. An introduction to the interactor matrix including its classification is given in Section 5.2. An estimation procedure for the unitary interactor matrix using the Markov parameters is discussed in Section 5.3 and unitary interactor estimation using closed-loop data is described in Section 5.5. Illustrative examples comparing the time-delay/interactor matrix estimated using the Markov parameters obtained using different methods are given in Section 5.6.

5.2 The Interactor Matrix

As mentioned earlier, time delays in multivariable systems are defined by the interactor matrix. Goodwin and Sin (1984)formally defined time delay in SISO systems

meaning there exists a scalar function $D(z) = z^{-d}$ such that

$$\lim_{z \to \infty} D(z) A(z^{-1}) B(z^{-1}) = k \tag{5.1}$$

where k is a nonzero scalar, d is the delay and A & B are the denominator and numerator respectively of the polynomial transfer function. They specified the multivariable delay structure in a similar way in terms of a polynomial matrix. The definition of the '*interactor matrix*', as given by Goodwin and Sin (1984)and Huang (1997), is:

Definition 1 *Given any $l \times l$, proper, rational polynomial transfer function matrix $G(z^{-1})$, there exists a unique, non-singular $l \times l$ lower left triangular polynomial matrix $D(z)$, known as the 'interactor matrix', satisfying:*

$$(i) \ det\, D(z) = z^r \tag{5.2}$$

$$(ii) \ \lim_{z \to \infty} D(z) G(z^{-1}) = \lim_{z \to \infty} G_0(z^{-1}) = K, \tag{5.3}$$

where r is an integer and represents the number of infinite zeros of $G(z^{-1})$, K is a non-singular matrix and $G_0(z^{-1})$ is the delay-free part of the transfer function matrix $G(z^{-1})$ and contains only the finite zeros. The 'interactor matrix' $D(z)$ can be expressed as

$$D(z) = D_0 z^d + D_1 z^{d-1} + \cdots + D_{d-1} z$$

where d is known as the order of the interactor matrix and the $D_i's$ $(i = 1, \cdots d-1)$ are coefficient matrices.

There are different forms of the interactor matrix such as diagonal, lower/upper triangular etc. Some of the special cases of the *interactor matrix* are:

(i) when $D(z) = z^d I$, the interactor is known as a *simple interactor matrix*. In this case, all the input-output pairs share the same number of delays and it is, therefore, easy to factor out the delay term from the delay-free part of the transfer function matrix,

$(ii)$when $D^T(z) D(z) = I$, the interactor is called a *unitary interactor matrix* that has advantages over other forms as discussed by Peng and Kinnaert (1992), Huang and Shah (1999).

This chapter focuses on the estimation of a unitary interactor matrix using the Markov block parameters obtained from the ARMarkov identification method.

Example 5.1 *(Huang and Shah 1999)*

Let us consider the plant transfer function,

$$G\left(z^{-1}\right) = \begin{bmatrix} \frac{z^{-1}}{1+z^{-1}} & \frac{z^{-1}}{1+2z^{-1}} \\ \frac{z^{-1}}{1+3z^{-1}} & \frac{z^{-1}}{1+4z^{-1}} \end{bmatrix}$$

The *unitary* interactor matrix for this transfer function is

$$D\left(z\right) = \begin{bmatrix} 0.5z + 0.5z^2 & 0.5z - 0.5z^2 \\ 0.5z^2 - 0.5z^3 & 0.5z^2 + 0.5z^3 \end{bmatrix}$$

and has the property $D^T\left(z\right) D\left(z\right) = I$. More details on the classification and properties of the *interactor matrix* can be found in the literature e.g. (Goodwin and Sin 1984, Shah *et al.* 1987, Peng and Kinnaert 1992, Huang and Shah 1999).

5.3 Estimation of the Unitary Interactor Matrix

Knowledge of the interactor matrix is important in designing high performance controllers such as minimum variance (MV) controllers and in closed-loop performance assessment. Earlier methods (Goodwin and Sin 1984, Wolowich and Falb 1976) of interactor estimation required *a priori* information of the process transfer function. Shah *et al.* (1987)suggested a method that does not require the above mentioned *a priori* information. It uses a linear combination of the first few Markov parameters of the process. The linear combination of the Markov block matrices is expressed as a polynomial matrix. Huang and Shah (1999)proposed the use of singular value decomposition (SVD) for the determination of the non-singularity of the polynomial matrix. The procedure is discussed in the following subsections (5.3.1-5.3.2).

5.3.1 Determination of the Order of the Interactor Matrix

A linear, time invariant (LTI) transfer function can be written in terms of Markov parameters as

$$G\left(z^{-1}\right) = \sum_{i=1}^{\infty} g_i z^{-i-1} \tag{5.4}$$

The condition (5.3) for the existence of the interactor matrix is written as

$$\lim_{z^{-1}\to 0} D(z)\,G(z^{-1}) = \lim_{z^{-1}\to 0} \left[D_0 z^d + D_1 z^{d-1} + \cdots + D_{d-1} z\right]\left[g_0 z^{-1} + g_1 z^{-2} + \cdots\right] = K \tag{5.5}$$

and can be expressed as a number of algebraic equations (Huang 1997, Huang and Shah 1999) that leads to the following matrix form

$$\left[\, D_{d-1},\ \cdots\ , D_0 \,\right] \begin{bmatrix} g_0 & 0 & 0 & \cdots & 0 \\ g_1 & g_0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_{d-2} & g_{d-1} & \cdots & \ddots & 0 \\ g_{d-1} & g_{d-2} & \cdots & \cdots & g_0 \end{bmatrix} = [K, 0, \cdots, 0] \tag{5.6}$$

$$\text{or} \quad \underline{D}\,\underline{G} = \underline{K} \tag{5.7}$$

where $\underline{G}$ is a block-Toeplitz matrix consisting of the first d process Markov parameters, $\underline{D}$ is the coefficient matrix of the interactor and K is a full rank matrix with $rank(K) = min(l, m)$. Existence of a solution of equation (5.7) which determines $\underline{D}$ depends on the order of the interactor matrix and the null space of $\underline{G}$. The $\underline{G}$ matrix needs to be expanded by adding Markov parameter blocks until certain conditions (see (Huang and Shah 1999)) are satisfied and $rank(\underline{G}) \geq rank(K)$. The order of the unitary interactor matrix is the maximum number of Markov block-parameters required to satisfy the above mentioned conditions regarding the singular value decomposition (SVD) of $\underline{G}$. Details of the SVD expansion and the procedure can be found in Huang and Shah (1999).

5.3.2 Estimation of the Interactor Matrix

Once the order, d, of the interactor is selected, a block matrix consisting of the first d Markov parameter blocks is written as

$$\Lambda = [G_0^T, G_1^T, \cdots G_d^T\,]^T \tag{5.8}$$

Once the block matrix, Λ is formed, the unitary interactor matrix, $D(q)$ can be factored out from Λ by following the procedure of Rogozinski *et al.* (1987)and Peng and Kinnaert (1992). As noted by Huang and Shah (1999), the plant transfer function matrix, $G(z^{-1})$ used in their procedure would be replaced by the first d Markov parameter matrices.

5.4 Markov Parameters from the ARMarkov Method for Interactor Estimation

As discussed in Section 5.3, Markov parameters are sufficient for interactor estimation even when *a priori* information on the process transfer function is not used or not available. Therefore, it is obvious that the better the estimation of the Markov parameters, the better the estimation of the interactor matrix will be since it uses a linear combination of those Markov parameters. Since the ARMarkov method produces Markov parameters directly and it was proved that the Markov parameters estimated by the ARMarkov method have better statistical qualities than the same parameters estimated by other simple linear regression methods, it is recommended that the interactor be estimated using the Markov parameters obtained from the ARMarkov method.

This chapter illustrates the application of the ARMarkov identification method and the Markov parameters estimated by this method to compute the interactor matrix. The interactor estimation procedure discussed in Section 5.3 was developed by Huang and Shah (1999)and is applied here. The interactors estimated using Markov parameters obtained by different methods are compared in Section 5.6 to show that the Markov parameters obtained from the ARMarkov method produce a better interactor matrix.

5.5 Interactor Matrix for Closed-Loop Systems

In industrial applications, closed-loop data is much easier to obtain and/or more readily available than open-loop data. Moreover, closed-loop data is preferred for on-line performance monitoring/assessment. Therefore, it is preferable to estimate the interactor matrix using closed-loop data. Huang and Shah (1999) showed that *the unitary interactor matrix is "feedback invariant" i.e. the linear combination of the Markov parameters of the process under open-loop and closed-loop conditions yields the same interactor matrix and the interactor matrix of the open-loop system can be estimated directly from closed-loop data.*

Lemma 5.1 *(Huang 1997, Huang* et al. *1997b) For a multivariable process as shown in Figure 5.1, the interactor (D_{cl}) of the closed-loop transfer function matrix, G_{cl} from W_t to Y_t is the same as the interactor (D) of the open-loop transfer function matrix, G_P.*

Proof.

$$\lim_{z^{-1}\to 0} DG_{cl} = \lim_{z^{-1}\to 0} DG_P(I + QG_P)^{-1} = K$$

where D is the interactor of the closed-loop transfer matrix, G_{cl} and since

$$\lim_{z^{-1}} D_{cl}G_P = K_{cl}$$

D_{cl} is also the interactor of the open-loop transfer matrix, G_P. See (Huang *et al.* 1997*b*) for more details. ■

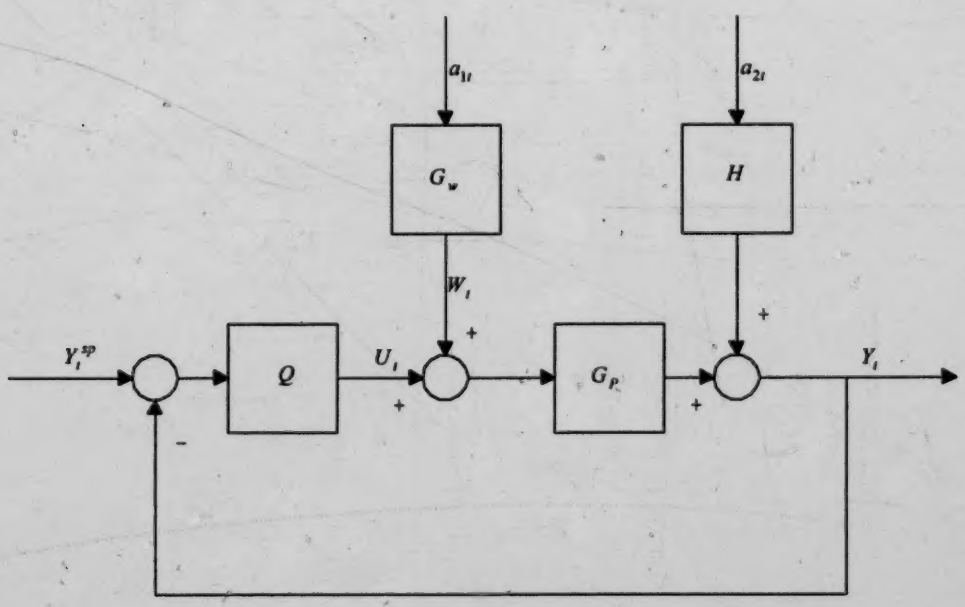


Figure 5.1: Closed-loop System.

The open-loop Markov parameters and the closed-loop Markov parameters of the same process are different, but the linear combination of the open or closed-loop Markov parameters leads to the same interactor matrix. In closed-loop systems,

identification using persistently exciting dither signals gives good estimates of the Markov parameters and in turn leads to a good estimate of the interactor matrix. In on-line performance monitoring/assessment, if step changes in the set points are made, they can be considered as the dither signal. In that case, correlation analysis may not be appropriate for the Markov parameters estimation. It was concluded that parametric modeling should give better Markov parameters and in turn better interactor estimation (Huang 1997). The ARMarkov method uses a overparameterized model that explicitly contains the required Markov parameters. Therefore, it is recommended that Markov parameters obtained by the ARMarkov identification method using closed-loop input-output data be used to estimate the interactor matrix. This is illustrated in Section 5.6.

5.6 Simulation Examples

This section describes simulations carried out to estimate the Markov parameters from process input-output data. The time delays/interactor matrices estimated using the Markov parameters determined by different linear regression methods are compared. Closed-loop input-output data as well as open-loop data are used to estimate the interactor matrix to illustrate the "feedback invariance" of the interactor matrix.

5.6.1 Open-Loop MIMO Systems

Example 5.2

This example, used by Huang (1997), illustrates the estimation of the interactor matrix of a simple 2x2 first order system with single delay in all the input-output pairs in the transfer function matrix.

The plant transfer function matrix is

$$G_p(z) = \begin{bmatrix} \frac{z^{-1}}{1-0.1z^{-1}} & \frac{z^{-1}}{1-0.1z^{-1}} \\ \frac{2z^{-1}}{1-0.3z^{-1}} & \frac{2z^{-1}}{1-0.4z^{-1}} \end{bmatrix} \tag{5.9}$$

and the disturbance transfer function matrix is

$$H = \begin{bmatrix} \frac{1}{1-0.5z^{-1}} & \frac{-0.6}{1-0.5z^{-1}} \\ \frac{0.5}{1-0.5z^{-1}} & \frac{1}{1-0.5z^{-1}} \end{bmatrix} \tag{5.10}$$

The unitary interactor matrices estimated by using the Markov parameters determined by different methods are shown in Table 5.1. The unitary interactor matrix estimated by using the Markov parameters determined by the ARMarkov method is exactly the same as in the actual process and the best estimate among all the methods used.

Table 5.1: Interactor Matrix for a 2x2 First order System

Method	Unitary Interactor Matrix
Actual Process	$\begin{bmatrix} -0.4472 * q & -0.8944 * q \\ 0.8944 * q^2 & -0.4472 * q^2 \end{bmatrix}$
ARMARKOV	$\begin{bmatrix} -0.4472 * q & -0.8944 * q \\ 0.8944 * q^2 & -0.4472 * q^2 \end{bmatrix}$
Correlation Analysis	$\begin{bmatrix} -0.4437 * q & -0.8962 * q \\ 0.8962 * q^2 & -0.4437 * q^2 \end{bmatrix}$
ARX	$\begin{bmatrix} -0.4469 * q & -0.8974 * q \\ 0.8974 * q^2 & -0.4469 * q^2 \end{bmatrix}$

The above example considered first order transfer functions with a unit delay in all the input-output pairs. For this reason, the interactor matrices estimated by using the Markov parameters determined by the different methods are close to the actual interactor matrix and close to one another. Processes with higher order transfer functions and higher order delays (also different delays in different input-output pairs) result in greater differences in the estimated parameters. The following example is concerned with such a process.

Example 5.3

Consider a 2×2 MIMO system with transfer function matrices

$$G_p(z) = \begin{bmatrix} \frac{0.3z^{-6}}{1-0.95z^{-1}} & \frac{0.0077z^{-1}+0.0212z^{-2}+0.0036z^{-3}}{1-1.9031z^{-1}+1.1514z^{-1}+0.2158z^{-3}} \\ \frac{z^{-6}}{1-0.7z^{-1}} & \frac{0.1091z^{-1}+0.07z^{-2}}{1-1.8044z^{-1}+0.2636z^{-2}} \end{bmatrix} \tag{5.11}$$

$$H = \begin{bmatrix} \frac{1}{1-0.95z^{-1}} & 0 \\ 0 & \frac{1}{1-0.99z^{-1}} \end{bmatrix} \tag{5.12}$$

The unitary interactor matrices estimated by different methods are shown in Table 5.2.

Table 5.2: Coefficients in the Interactor Matrix

Interactor structure: $\begin{bmatrix} m1 * q^1 + \cdots + m5 * q^5 & p5 * q^1 + \cdots + p1 * q^5 \\ p1 * q^2 + \cdots + p5 * q^6 & m5 * q^2 + \cdots + m1 * q^6 \end{bmatrix}$

	m1	m2	m3	m4	m5	p1	p2	p3	p4	p5
Actual	.0680	.1957	.0897	.1087	.0609	.0043	.0203	.0353	.0517	.9639
ARMark	.0722	.1939	.0896	.1146	.0539	.0040	.0196	.0360	.0524	.9637
Correl	.0876	.1914	.0550	.0704	.1268	.0115	.0321	.0281	.0443	.9632
ARX	.0596	.1987	.1127	.0036	.1109	.0068	.0229	.0202	.0361	.9642

The differences between each coefficient in the actual unitary interactor matrix and the unitary matrix estimated by different methods are plotted in Fig. 5.2 so that the accuracy of the estimated parameters can be compared more easily.

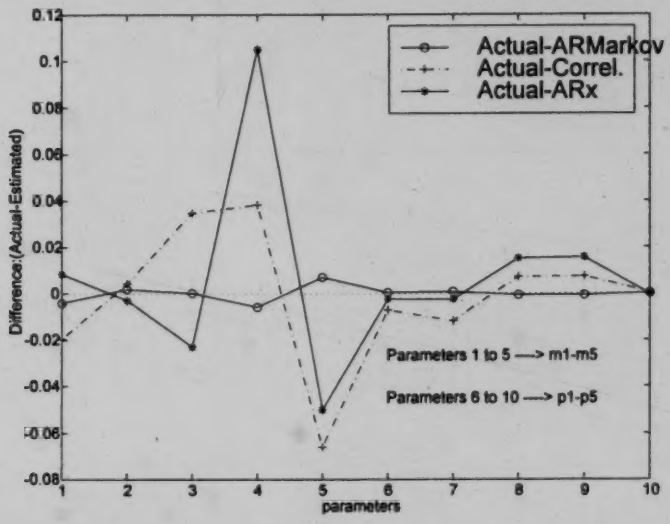


Figure 5.2: Errors in the estimated coeffiecients of the interactor

It can be clearly seen that the coefficients estimated by the ARMarkov method have the smallest error.

5.6.2 Closed-loop Systems

Since closed-loop data are more readily available and/or easier to collect in practical/industrial applications, the ARMarkov method was also applied to estimate the Markov parameters from closed-loop input-output data. A simple closed-loop system

is shown in Fig. 5.1. The purpose here was not to design a controller but to show that the open-loop and closed-loop unitary interactor matrices are the same as proved theoretically by Huang (1997). Therefore, a simple proportional controller was used. The next example used the same first order plant and disturbance transfer functions as in Example 5.2 for the open-loop case.

Example 5.4

The transfer function matrices used are

$$G_p(z) = \begin{bmatrix} \frac{z^{-1}}{1-0.1z^{-1}} & \frac{z^{-1}}{1-0.1z^{-1}} \\ \frac{2z^{-1}}{1-0.3z^{-1}} & \frac{2z^{-1}}{1-0.4z^{-1}} \end{bmatrix}, \; H = \begin{bmatrix} \frac{1}{1-0.5z^{-1}} & \frac{-0.6}{1-0.5z^{-1}} \\ \frac{0.5}{1-0.5z^{-1}} & \frac{1}{1-0.5z^{-1}} \end{bmatrix} \tag{5.13}$$

$$Q = \begin{bmatrix} 0.4 & 0 \\ 0 & 0.3 \end{bmatrix}, \; G_w = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \; Y_t^{sp} = 0 \tag{5.14}$$

The unitary interactor matrices estimated by the different methods using closed-loop input-output data are given in the Table 5.3. In this example also, the ARMarkov method gives the best match with the actual process and the interactor matrix is almost the same as in the open-loop case shown in Table 5.1. Other simulations (shown in Chapter 2) show that the disturbance dynamics have less effect on the ARMarkov method than on the other methods.

Table 5.3: Interactor Matrix from Closed-loop Data

Method	Unitary Interactor Matrix
Actual Process	$\begin{bmatrix} -0.4472 * q & -0.8944 * q \\ 0.8944 * q^2 & -0.4472 * q^2 \end{bmatrix}$
ARMARKOV Method	$\begin{bmatrix} -0.4469 * q & -0.8945 * q \\ 0.8945 * q^2 & -0.4469 * q^2 \end{bmatrix}$
Correlation Analysis	$\begin{bmatrix} -0.4494 * q & -0.8933 * q \\ 0.8933 * q^2 & -0.4494 * q^2 \end{bmatrix}$
ARX	$\begin{bmatrix} -0.4534 * q & -0.8913 * q \\ 0.8913 * q^2 & -0.4534 * q^2 \end{bmatrix}$

5.7 Conclusions

- The time delay/interactor matrix estimated from the Markov parameters obtained using the ARMarkov identification method is closer to the actual time delay/interactor matrix than comparable results obtained using other linear regression methods such as ARX, FIR.

- The Interactor matrix is theoretically feedback invariant. Simulation examples using the ARMarkov identification method show that the interactor matrix calculated from parameters estimated using closed-loop data is almost identical to the one determined from open-loop data.

Part II

Model Predictive Controller Design

This part of the thesis describes the design of MPC's, referred to as ARM-MPC's that use the ARMarkov model developed in Part I. The ARM-MPC is formulated in Chapter 8 based on an extended input-output ARMarkov model. It is flexible enough to include the characteristics of the two widely used MPC's, GPC and DMC, as special cases. The ARM-MPC does not require the solution of Diophantine equations as in GPC and does not need as large a number of step response coefficients or prediction horizon as in DMC. In Chapter 9 it was shown that the observer polynomial in ARM-MPC plays the same role as in GPC. However, the observer was designed without the solution of Diophantine equations and is independent of the process model which leads to advantages such as the use of different disturbance horizons and independent tuning for servo and regulatory control. State space models can be extended more easily to MIMO systems than input-output models and also are often more convenient for simulation and theoretical analysis. Therefore, a dual-model, state space model is derived from the estimated input-output ARMarkov model. It is equivalent to the input-output model but has a different structure than the classical state space formulation discussed in Chapter 9. The state space ARM-MPC designed in Chapter 9 using a classical state space model generated from the consistent Markov parameters identified during the input-output ARMarkov model identification in Part I has the classical $[A, B, C, D]$ state space structure. All the classical theories and properties of state space MPC are, therefore, applicable to this ARM-MPC.

Closed-loop performance of different ARM-MPC's is discussed in Part III.

Chapter 6

Design and Formulation of an ARMarkov Model-based Predictive Controller (ARM-MPC)[1]

6.1 Introduction

Model Predictive Controllers (MPC) typically use the step or impulse response coefficients of the process model to build the '*dynamic matrix*' used in the output prediction equation. The Generalized Predictive Controller (GPC) of Clarke *et al.* (1987*b*) uses an input-output model with an ARIMAX structure, builds the '*dynamic matrix*' using the Markov parameters and requires the solution of two Diophantine equations (Clarke *et al.* (1987*b*), Bitmead *et al.* (1990)). Dynamic Matrix Control (DMC), the most widely used MPC in industry, uses a simple step-response model for the output prediction (Cutler and Ramaker (1980)).

Unlike GPC, Dynamic Matrix Control does not require the Diophantine solutions, but needs a large number of step-response coefficients. Li *et al.* (1989) showed that MPC using step-response models can be put in state-space form. Lee *et al.* (1994) presented a state-space model in terms of step-response coefficients for systems with stable and/or integrating dynamics and extended the conventional MPC to handle stochastic and white noise disturbances without solving a large-order Riccati equation. Morari and Lee (1991) identified some deficiencies of MPC by comparing it with Linear Quadratic Gaussian control concepts. Camacho and Bordons (1999)

[1]This chapter was presented at the ADCHEM 2000 conference, June 2000 and is a part of the paper to be published in the *Journal of Process Control*

present techniques for implementing GPC in industrial processes. Banerjee (1996) formulated a Markov-Laguerre model based predictive controller that uses a non-parametric model for the fast dynamics of the process and a parametric Laguerre model to approximate the slower dynamics of the process. Qi (1997) formulated an MPC based on a dual model (step response plus autoregressive).

Parametric model based predictive controllers, e.g. GPC, tend to be aggressive, less robust and require careful estimation of the parametric model and time delay. They often include disturbance models which increase the flexibility of tuning. On the other hand, non-parametric model-based controllers, e.g. DMC, are less aggressive, use simple step response models but require large number of step responses. Both parametric and non-parametric model-based predictive controllers have advantages and disadvantages.

In all the predictive controller designs, the *dynamic matrix* plays important role in the formulation of the controller gain matrix. The *dynamic matrix* is constructed using the first N_2 step/impulse response coefficients (or Markov parameters) where N_2 is the prediction horizon. The better the estimation of the coefficients, the better the representation of the actual process and in turn the more accurate the controller gain matrix. For processes with unusual step responses in the fast dynamics, a parametric model can not always capture the actual dynamics of the process. In these cases, direct estimation of the Markov parameters or step response coefficients is very important. Specification of the time delay plays an important role in the parametric model based predictive controller formulation.

The predictive controller developed in this chapter uses an ARMarkov model representation that is a combination of non-parametric and parametric models. The initial or fast dynamics are defined by Markov parameters estimated explicitly using a standard least squares algorithm and the slower dynamics are approximated by an ARIMAX model structure. The ARMarkov model has the same form as an ARIMAX model except that it explicitly contains more than one Markov parameter (Akers and Bernstein (1997)). Details of the ARMarkov approach are described in Chapter 2. The estimated Markov parameters are consistent in the presence of disturbances as presented in Chapter 4 and shown by Kamrunnahar *et al.* (2000). In the residual model structures of the dual-model (Qi 1997) and Markov-Laguerre (Banerjee 1996)

model formulations, the continuity of the dual responses is not obvious and the parameter consistency is not as straightforward as in the ARMarkov identification.

An ARIMAX structure is introduced to model the slow dynamics and the disturbances in Section 6.2 and the resulting model is called the *extended* ARMarkov model. A predictive controller is then formulated in Section 6.3 by using the extended ARMarkov model and an approach similar to that used by other predictive controller designs such as GPC (Clarke *et al.* (1987*b*)) and DMC (Cutler and Ramaker (1980)). In Section 6.4, it is shown that the proposed predictive controller, referred to as ARM-MPC, is equivalent to GPC and DMC when the number of Markov parameters, μ, used in the model is 1 and $N_2 + 1$ respectively. For $1 < \mu \leq N_2 + 1$, the proposed controller blends the characteristics of GPC and DMC. The ARM-MPC is illustrated through simulation examples in Section 6.5.

6.2 The Extended ARMarkov Model

The extended ARMarkov model with a general disturbance model is defined as

$$\begin{aligned} y(k) &= -\sum_{j=1}^{n} \alpha_j y(k-\mu-j+1) + \sum_{j=0}^{\mu-1} h_j u(k-j) + \sum_{j=1}^{n} \beta_j u(k-\mu-j+1) \\ &\quad + \frac{C(z^{-1})}{\Delta}\epsilon(k) \qquad (6.1) \\ &= -\sum_{j=1}^{n} \alpha_j y(k-\mu-j+1) + \sum_{j=0}^{\mu} h_j u(k-j) + \sum_{j=2}^{n} \beta_j u(k-\mu-j+1) \\ &\quad + x(k) \qquad (6.2) \end{aligned}$$

where C is a polynomial in z^{-1}, $x(k)$ is the disturbance or residual and $\Delta = 1 - z^{-1}$. For chemical process applications, it is very common to assume random step-type disturbances, often termed Type-I disturbances. Therefore, in this chapter, the ARMarkov model is modified to accommodate Type-I disturbances by putting $C = 1$ in (6.1). The extended ARMarkov model and the extended ARMarkov identification method with a general disturbance model ($C \neq 1$) is discussed in Chapter 7.

The extended ARMarkov model with Type-I disturbances is defined as

$$y(k) = -\sum_{j=1}^{n} \alpha_j y(k-\mu-j+1) + \sum_{j=0}^{\mu-1} h_j u(k-j) + \sum_{j=1}^{n} \beta_j u(k-\mu-j+1) + \frac{1}{\Delta}\epsilon(k) \tag{6.3}$$

Writing the ARMarkov regressor vector, ϕ_μ for equation (6.3) as

$$\phi_\mu(k) = [y(k-1), \Delta y(k-\mu), \cdots, \Delta y(k-\mu-n+1), \Delta u(k), \cdots, \Delta u(k-\mu-n+1)]^T \tag{6.4}$$

the process output can be expressed as

$$y(k) = W_\mu \phi_\mu(k) + \epsilon(k) \tag{6.5}$$

where the weighting matrix is given by

$$W_\mu = [1, -A_\mu, h_0, \cdots, h_{\mu-1}, B_\mu] \tag{6.6}$$

$$A_\mu = [\alpha_1, \cdots, \alpha_n] \in R^{1\times n} \tag{6.7}$$

$$B_\mu = [\beta_1, \cdots, \beta_n] \in R^{1\times n} \tag{6.8}$$

The regressor vector in (6.4) differs from the regressor vector in (2.19) of Chapter 2 by including $y(k-1)$ and using the remaining input-output deviation variables instead of the absolute variables. The parameters vector W_μ in (6.6) also differs from (10.3) by including an extra 1. The minimum variance prediction of the output is

$$\begin{aligned} \widehat{y}(k) &= -\sum_{j=1}^{n} \alpha_j \Delta y(k-\mu-j+1) + \sum_{j=0}^{\mu-1} h_j \Delta u(k-j) \\ &\quad + \sum_{j=1}^{n} \beta_j \Delta u(k-\mu-j+1) + y(k-1) \end{aligned} \tag{6.9}$$

The output error, $\varepsilon(k)$ is defined as

$$\varepsilon(k) = y(k) - \hat{y}(k) \tag{6.10}$$

The estimated parameter vector, $\hat{W}_\mu$ can be determined by using a standard least-squares algorithm to minimize the cost function $J = \frac{1}{N}\sum_{k=1}^{N} \frac{1}{2}\varepsilon^2(k)$ where N is the total number of data points.

Note: Due to the use of differenced data in the regressor vector in (6.4), information on the low frequency dynamics of the process may not be correctly estimated.

Therefore, the above identification method with a type-I disturbance model is recommended for process identification when there is a trend in the input/output data.

Note, however, that the type-I disturbance model should be used during the MPC controller design to obtain offset-free setpoint tracking as discussed in the next section.

6.3 Predictive Control Design Using the Extended ARMarkov Model

As shown below, a Model Predictive Controller can be formulated using the extended ARMarkov model described in the previous section. The resulting controller is referred to as ARM-MPC.

In predictive control, it is necessary to compute the future output predictions, $y(k+i)$, for $i = 1, \cdots, N_2$ where N_2 is the prediction horizon. Assuming a single delay due to discretization, the output prediction equation, using the extended ARMarkov model in (6.3) can be written as

$$\begin{aligned} y(k+i) &= y(k+i-1) - \sum_{j=1}^{n} \alpha_j \Delta y(k-\mu-j+1+i) + \sum_{j=1}^{\mu-1} h_j \Delta u(k-j+i) \\ &\quad + \sum_{j=1}^{n} \beta_j \Delta u(k-\mu-j+1+i) + \epsilon(k+i) \end{aligned} \tag{6.11}$$

The last term, consisting of future white noise, is independent of the available response at time k. The minimum variance prediction is independent of the white noise and is given by

$$\begin{aligned} \widehat{y}(k+i) &= \widehat{y}(k+i-1) - \sum_{j=1}^{n} \alpha_j \Delta y(k-\mu-j+1+i) + \sum_{j=1}^{\mu-1} h_j \Delta u(k-j+i) \\ &\quad + \sum_{j=1}^{n} \beta_j \Delta u(k-\mu-j+1+i) \end{aligned} \tag{6.12}$$

After separating the prediction equation into the information available at time k and the future information needed, the output prediction can be written as

$$\widehat{y}(k+i) = G_i(z^{-1})\Delta u(k+i-1) + \widehat{y}(k+i|k) \tag{6.13}$$

where G_i is a polynomial in z^{-1} containing the first i step response coefficients s_i, i.e.

$$G_i = s_1 z^0 + s_2 z^{-1} + \cdots + s_i z^{-(i-1)} \tag{6.14}$$

where $s_i = h_i + s_{i-1}$. The $\Delta u(k+i-1)$ term in (6.13) is the present/future incremental control action. $\widehat{y}(k+i|k)$ is the output prediction which depends only on the information available at time k and usually referred to as the 'free response'. Note that $\beta_1 = h_\mu$ in (6.12). The 'free response' can be written as

$$\begin{aligned}\widehat{y}(k+i|k) &= y(k) - \sum_{j=1}^{n} \alpha_j y(k-\mu-j+1+i) + \sum_{j=1}^{n} \alpha_j y(k-\mu-j+1) \\ &+ \sum_{j=1}^{\mu+n-1-i} s_{i+j}\Delta u(k-j) + \sum_{j=\mu+n-i}^{\mu+n-1} s_{\mu+n-1}\Delta u(k-j) \\ &- \sum_{j=1}^{\mu+n-2} s_j \Delta u(k-j-1) \end{aligned} \tag{6.15}$$

and can be simplified to

$$\widehat{y}(k+i|k) = y(k+i|k-1) + s_{i+1}\Delta u(k-1) \quad \text{for } i = 1, \cdots, \mu-1 \tag{6.16}$$

$$\begin{aligned}\widehat{y}(k+i|k) &= y(k+i-1|k-1) - \alpha_1 y(k) + \alpha_n y(k-n) \\ &- \sum_{j=1}^{n-1} (\alpha_{j+1} - \alpha_j) y(k-\mu-j+1+i) \\ &+ s_{i+1}\Delta u(k-1) + \sum_{j=3}^{n} \beta_j \Delta u(k-\mu-j+1+i) \quad \text{for } i = \mu \end{aligned} \tag{6.17}$$

The 'free response' vector is defined as

$$f = [\widehat{y}(k+1|k), \widehat{y}(k+2|k), \cdots, \widehat{y}(k+N_2|k)]^T \tag{6.18}$$

and the future control vector as

$$\widehat{\mathbf{u}} = [\Delta u(k), \Delta u(k+1), \cdots, \Delta u(k+M-1)]^T \tag{6.19}$$

where M is the control horizon. The predicted output vector is defined as

$$\widehat{\mathbf{Y}} = [\hat{y}(k+1), \hat{y}(k+2), \cdots, \hat{y}(k+N_2)]^T \tag{6.20}$$

so that

$$\widehat{\mathbf{Y}} = G\widehat{\mathbf{u}} + f \tag{6.21}$$

$$\text{where } G = \begin{bmatrix} s_1 & 0 & \cdots & 0 & 0 \\ s_2 & s_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ s_M & s_{M-1} & \cdots & \cdots & s_1 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ s_{N_2} & s_{N_2-1} & \cdots & \cdots & s_{N_2-M+1} \end{bmatrix} \tag{6.22}$$

The prediction equation in (6.21) is similar to the one used in GPC, DMC etc. The G matrix in (6.22) consists of the step response coefficients and is known as the '*Dynamic Matrix*' in Model-based Predictive Controller (MPC) design.

The future control moves, $\widehat{\mathbf{u}}$ can be estimated by minimizing a cost function such as,

$$J = E^T Q E + \lambda \widehat{\mathbf{u}}^T R \widehat{\mathbf{u}} \tag{6.23}$$

where Q and λR are the output and input weightings respectively, E is the error between the predicted output and set-point. This is now a linear control problem and can be solved easily. The unconstrained solution of the future control vector is

$$\widehat{\mathbf{u}} = (G^T Q G + \lambda R)^{-1} G^T Q (r - f) \tag{6.24}$$

where the reference signal, $\mathbf{r} = [r(k+1) \quad r(k+2) \quad \cdots \quad r(k+N_2)]^T$ is the set-point or target vector.

Remark 6.1 *The Markov parameters determined by the ARMarkov identification method were shown by Kamrunnahar* et al. *(2000) and in Chapter 5 to be superior in terms of consistency, variances/covariances and confidence bounds to the Markov parameters determined indirectly using other linear regression methods. The controller gain matrix in ARM-MPC is defined as*

$$K_{mpc} = (G^T Q G + \lambda R)^{-1} G^T Q$$

where Q and λR are user defined and fixed. 'Dynamic Matrix' G is derived from ***consistent*** *Markov parameters and therefore in the limit as the Markov parameters*

converge, the 'Dynamic Matrix' G is error free. In other words, the 'Dynamic Matrix' constructed by using the Markov parameters identified using the ARMarkov method, in the limit, leads to an error free controller gain matrix, K_{mpc}.

Now consider the following special cases.

- $\mu = N_2$ i.e. the number of Markov parameters equals the prediction horizon. In this case, the output prediction can be easily separated into the future response and the 'free response'. The 'free response', $\widehat{y}(k+i|k)$ is the same as (6.15) except that μ is replaced by N_2.
- $\mu > N_2$ i.e. the number of Markov parameters is greater than the prediction horizon. In this case, the free response is the same as (6.15).
- $\mu < N_2$ i.e. the number of Markov parameters is less than the prediction horizon. The ARMarkov model cannot be used directly in this case since only μ Markov parameters are available and $N_2 > \mu$ Markov parameters are needed to build the '*dynamic matrix*'. In this case, solutions to Diophantine type equations are required as in GPC for the remaining $(N_2 - \mu)$ Markov parameters.

Remark 6.2 *For $\mu \geq N_2$ (the prediction horizon), the output prediction is easily separated into the future and the 'free response' without the solution of Diophantine equations.*

6.4 Special Cases of ARM-MPC

The ARMarkov model in (6.3) can be written as

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + \frac{C(z^{-1})}{\Delta}\epsilon(k) \tag{6.25}$$

where

$$A = 1 + \alpha_1 z^{-\mu} + \cdots + \alpha_n z^{-(\mu+n-1)} \tag{6.26}$$

$$B = h_1 z^{-0} + \cdots + h_{\mu-1} z^{-(\mu-1)} + \beta_1 z^{-\mu} + \cdots + \beta_n z^{-(\mu+n-1)} \tag{6.27}$$

$$C = 1 \tag{6.28}$$

The basic Generalized Predictive Control (GPC) of Clarke *et al.* (1987*b*) uses an ARIMAX process model of the form

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + \frac{C(z^{-1})}{\Delta}\epsilon(k) \qquad (6.29)$$

where A, B, C are polynomials in z^{-1}. Therefore, the extended ARMarkov model used in the ARM-MPC has the same form as the ARIMAX model used in GPC. However, the number of parameters is different in the two models.

The optimal prediction of $y(k+i)$, as described by Bitmead *et al.* (1990), is

$$\widehat{y}(k+i) = G_i(z^{-1})\Delta u(k+i-1) + \widehat{y}(k+i|k) \qquad (6.30)$$

where $\widehat{y}(k+i|k)$ is the 'free response' prediction,

$$\widehat{y}(k+i|k) = \Gamma_i(z^{-1})u^f(k-1) + F_i(z^{-1})y^f(k) \qquad (6.31)$$

$$u^f(k) = C^{-1}\left(z^{-1}\right)\Delta u\left(k\right) \qquad (6.32)$$

$$y^f(k) = C^{-1}\left(z^{-1}\right)y\left(k\right) \qquad (6.33)$$

and Γ_i, F_i are the solutions to the following two Diophantine equations as shown in Clarke *et al.* (1987*b*) and Bitmead *et al.* (1990).

$$\begin{aligned} C(z^{-1}) &= E_j(z^{-1})A(z^{-1})\Delta + q^{-j}F_j\left(z^{-1}\right) \\ E_j(z^{-1})A(z^{-1}) &= G_j(z^{-1})C(z^{-1}) + q^{-j}\Gamma_j(z^{-1}) \end{aligned}$$

The prediction equation (6.13) using the extended ARMarkov model and the GPC prediction in (6.30) have the same form. The '*dynamic matrix* ' G is the same for both the prediction formulations. However, as shown above in Section 6.3, the ARM-MPC solution, unlike the GPC solution, does not require solution of a Diophantine equation when $\mu \geq N_2$.

The limiting cases of the ARM-MPC are derived as follows.

6.4.1 GPC structure:

Assuming type-I disturbances and $\mu = 1$, equations (6.26-6.28) can be written as

$$\begin{aligned} A &= 1 + \alpha_1 z^{-1} + \cdots + \alpha_n z^{-n} \\ B &= \beta_1 z^{-1} + \cdots + \beta_n z^{-n} \\ C &= 1 \end{aligned}$$

which is exactly the same as the ARIMAX model used in the GPC design with $C = 1$. Hence the designed ARM-MPC with $\mu = 1$ is mathematically equal to GPC.

6.4.2 DMC structure:

With $\mu = N_2 + 1$ and $n = 0$, equations (6.26-6.28) can be written as

$$\begin{aligned} A &= 1 \\ B &= h_1 z^{-0} + \cdots + h_{N_2} z^{-N_2} \\ C &= 1 \end{aligned}$$

which is exactly the same model used in DMC (Cutler and Ramaker 1980).

Remark 6.3 *From the above analysis, it is clear that using the ARM-MPC, one can implement GPC, DMC or a blend of the two by appropriate choice of the design parameters, μ and n. Other MPC's based on dual models, e.g. (Qi 1997) or Markov-Laguerre model (Banerjee 1996), can be interpreted for the limiting case of DMC, but the GPC structure is not clear and the combination is not as obvious as the ARM-MPC. Moreover, those non-ARM dual-models need larger number of Markov parameters and the continuity of the output response at the point where the two models join is not clear.*

6.5 Simulation Examples Using an ARM-MPC

The theory developed in the previous sections is illustrated through simulation examples in this section. The ARMarkov identification procedure and predictive controller design are used to control the following third order process

$$G(z) = \frac{0.0077z^{-1} + 0.0212z^{-2} + 0.0036z^{-3}}{1 - 1.9031z^{-1} + 1.1514z^{-2} - 0.2158z^{-3}} \tag{6.34}$$

A Type-I disturbance model is used in the ARMarkov model. The model parameters are determined using the model structure in (6.3). The ARMarkov model order is assumed to be $n = 3$ and the number of Markov parameters in the model is $\mu = 10$. The estimated parameters are then used in a predictive controller design having the structure in (6.13). The controller is designed as in section 6.3. The tuning

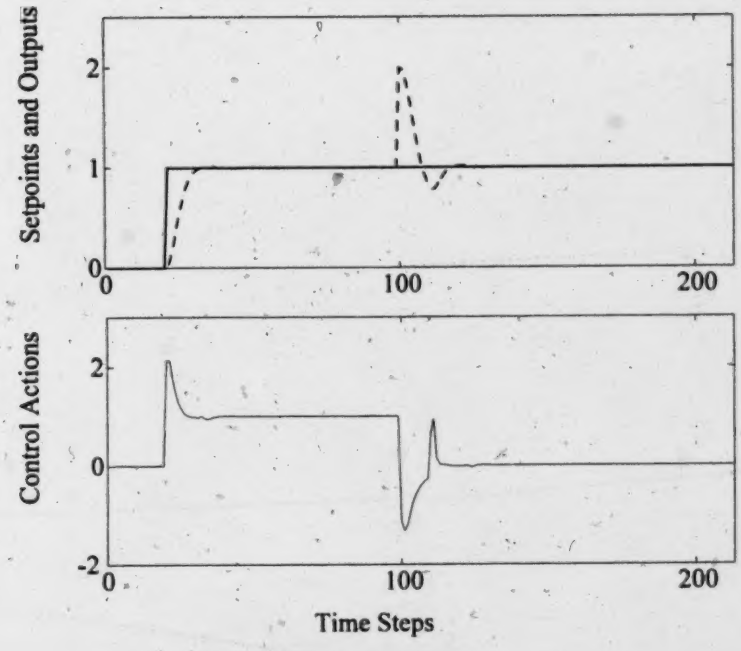


Figure 6.1: Step Tracking and Disturbance Rejection using GPC ARM-MPC with a Type-I Disturbance Model. A unit step disturbance was added at time 100.

parameters for the controller design are as follows: $N_2 = 10$, $M = 1$, $Q = R = I$ and $\lambda = 0.0$.

The closed-loop responses to set-point changes and disturbances are shown in Figure 6.1. The ARM-MPC with a Type-I disturbance model gives satisfactory responses to both set-point changes and disturbances. The responses are compared with GPC and DMC step-tracking and disturbance rejection in Figure 6.2. GPC is aggressive (note large span of axis for control action) and DMC needs a large number of step response coefficients and a large prediction horizon for practical applications. However, as expected, the ARM-MPC responses have characteristics which are a blend of GPC and DMC responses. The performance of GPC, DMC and ARM-MPC in terms of closed-loop settling time and input variance is compared in Table 6.1. GPC and ARM-MPC settling times are essentially the same whereas DMC takes almost twice as long to settle. The closed-loop time constants (rise times) are approximately the same for the three controllers but DMC has more overshoot in the servo response

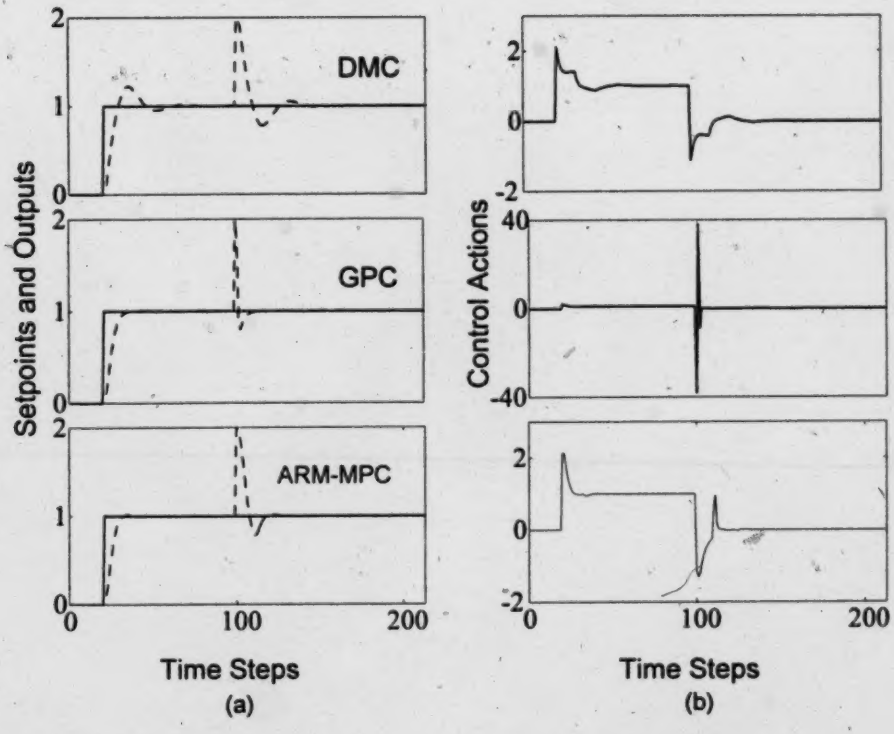


Figure 6.2: Comparison of Responses from DMC, GPC and ARM-MPC with a Type-1 disturbance model. In (a) 'solid'→ setpoints and 'dash'→ outputs. A unit step disturbance was added at time 100.

and overshoot/undershoot in the disturbance rejection. A larger prediction horizon in DMC gives overshoot-free response (not shown) which in turn makes the time constant larger than those by GPC and ARM-MPC. GPC disturbance rejection is faster than both ARM-MPC and DMC since GPC without a $T-$filter is very aggressive. This is reflected in the control action i.e. GPC results in high control action which may not be acceptable for practical applications. As a trade-off between DMC and GPC, the proposed ARM-MPC gives adequate disturbance rejection with an acceptable servo response. In terms of aggressiveness, it lies in between GPC and DMC. A measure of the performance of the predictive controller is defined as

$$J = \sum_{i=1}^{N_1} \left(Y_{sp}(i) - Y_{out}(i)\right)^T \left(Y_{sp}(i) - Y_{out}(i)\right) + \lambda u(i)^T u(i) \qquad (6.35)$$

where N_1 is the total simulation time. The performance costs for the above three

controllers are compared in Table 6.1. Clearly, the ARM-MPC performance cost lies in between the costs of GPC and DMC showing a trade-off between the GPC and DMC.

Table 6.1: Close-loop Performance of Different Controllers with $C = 1$

	GPC	DMC	ARM-MPC
Settling Time	11.5	24	11
Time Constant	6.0	6.5	6.0
var(u)	14.47	0.32	0.33
Performance Cost, J	5.01	8.43	7.32

6.5.1 Comparison of Linear Systems

The preceding comparison of DMC, GPC and ARM-MPC involves the performance of linear systems. In general, if the process model, controller and process inputs are identical, the closed-loop responses should be identical. However, in the above comparison there are a number of factors that result in different system responses. For example:

- DMC used the first ten step response coefficients. ARM-MPC used the first ten Markov (impulse) parameters PLUS a third-order ARX model that was sufficient to accurately model the residual process response. In other words, DMC used a truncated model and ARM-MPC used a "complete" model.
- The dynamic matrices for all three controllers were identical. However, the calculation of the control action depends strongly on design parameters such as the output horizon, N_2 control horizon, M and on the "free response" (which depends on the model used).
- In GPC the denominator, A, of the process input-output model automatically appears in the denominator of the noise model which makes the control response much more aggressive. Also there is interaction between the process noise/disturbance responses because the noise and model parameters are mixed in the Diophantine identities. (Saudagar 1995)

6.6 Experimental Evaluation of ARM-MPC on a Pilot Scale Process

An MPC using a general state space model developed from the Markov parameters obtained using the ARMarkov method was implemented on a pilot scale Continuous Stirred Tank Heater (CSTH) in the Computer Process Control (CPC) laboratory in the Chemical & Materials Engineering department at the University of Alberta. The CSTH is described in Chapter 2 and a schematic diagram of the process is shown in Figure 2.3. Water level in the tank and the water temperature were selected as the two controlled variables and the valve openings (manipulating cold water flow and steam flow) were selected as the two manipulated variables. The process considered is a 2×2 MIMO system. However, the water level in the tank varies only with the inlet water flow and is invariant to the steam flow rate.

The main purpose here was to complete an experimental application of the proposed ARM-MPC on a real process. An ARM-MPC was designed as described in Section 6.3 using an ARMarkov model developed for the above process using real time, experimental open-loop input-output data and the ARMarkov identification method described in Chapter 2. The ARMarkov model order was $n = 2$ and the number of Markov parameters in the model was assumed to be $\mu = 26$. The controller parameters used were as follows: $N_2 = 10$, $M = 1$, $Q = R = I$ and $\lambda = 0.0$. The inputs and outputs were sampled every four seconds. Set-points, outputs and manipulated variables were measured as currents signals. Servo responses from the CSTH are shown in Figure 6.3.

It is very clear from Figure 6.3 that the designed ARM-MPC tracks the set-point changes very satisfactorily. Note, however, that the inlet water flow rate was kept constant while controlling the water temperature to make the process simpler and observe the effects of each manipulated variable on the process output. The major process noise disturbances were due to incomplete stirring/mixing of the water.

6.7 Conclusions

- The ARMarkov model is extended to include Type-I disturbance models.

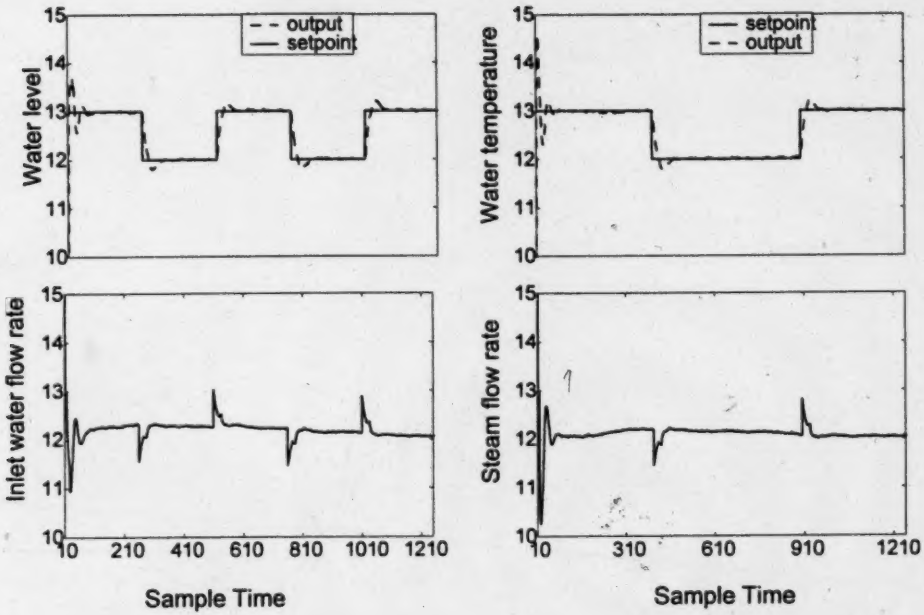


Figure 6.3: Responses from a pilot scale CSTH using ARM-MPC.

- Output predictions based on the extended model are shown to have the same $\hat{y} = G\hat{\mathbf{u}} + f$ form used by predictive controllers such as GPC and DMC.
- The proposed ARM-MPC controller is equal to GPC when the number of Markov parameters, $\mu = 1$ and to DMC when $\mu = N_2 + 1$. For $1 < \mu \leq N_2 + 1$, ARM-MPC combines the characteristics of GPC (aggressive) and DMC (conservative).
- The Markov parameters determined by the ARMarkov method are statistically consistent and hence the '*dynamic matrix*' formed with the Markov parameters from the ARMarkov identification leads to better control.
- Experimental results showing the set-point tracking performance of the proposed ARM-MPC on a physical process (pilot scale CSTH) are satisfactory.

Chapter 7

Disturbance Prediction in the ARM-MPC[1]

7.1 Introduction

One of the distinguishing features of Generalized Predictive Controller (GPC) (Clarke *et al.* 1987*b*) compared to Dynamic Matrix Control (DMC) (Cutler and Ramaker 1980) is the disturbance model. The role of the disturbance polynomial, $C(z^{-1})$, also known as the prefilter for parameter estimation or the observer polynomial, $T(z^{-1})$ has been explored by many researchers (Astrom and Wittenmark 1984, Wahlberg and Ljung 1986, Clarke *et al.* 1987*b*, Doyle and Stein 1979) for the 'optimality of predictions', 'disturbance rejection' and 'robustness of model-based controllers'. Mohtadi (1988)discussed the robust stability issue of GPC using the $C(z^{-1})$ or $T(z^{-1})$ polynomial and pointed out the strong 'need for having (1) a prefilter for parameter estimation and (2) an observer polynomial in the control law' in the applications of GPC algorithm to industrial processes.

Successful identification of the C polynomial is not always possible for real processes. Moreover, in MPC output predictions, the C polynomial determines the dynamics of the observer equivalent to a Kalman filter. Therefore, it is usually suggested that instead of using the identified C polynomial, a design polynomial $T(z^{-1})$ be used to specify the observer dynamics (Clarke *et al.* 1987*b*, Bitmead *et al.* 1990). When $T = C$, the output predictions asymptotically reduce to the (minimum variance) 'optimal' predictions. The output predictions and the solutions to the GPC

[1]Material in this chapter forms a part of the paper published in the Journal of Process Control.

control problem are obtained in terms of the solutions of Diophantine equations. The Diophantine equations can be solved recursively to save computational effort.

Due to the structure of the CARIMA model considered in GPC, the output predictions or more precisely the 'free response' predictions cannot be separated from the predictions based on the plant model and based on the disturbance model. As a result, disturbances can not be predicted independent of the plant model and hence the controller can not be tuned independently for 'servo' and 'regulatory' control without designing a two-degree of freedom controller.

Saudagar (1995)introduced a method called the Separated Diophantine Predictor (SDP) that separates the noise model from the plant model in the prediction equation. This separation is obtained via the solution of two Diophantine equations and the end result is independent tuning for regulatory control. However, the resulting observer polynomial, F_i obtained through the solution of the Diophantine equations is dependent on the plant model polynomial, A.

In this chapter, the ARMarkov model is extended to include a disturbance polynomial, C equivalent to that included in the CARIMA model used in GPC. The noise/disturbance model can be estimated using plant data as described in Section 7.2 and then can be used for the ARMarkov model based controller (ARM-MPC) design. Alternatively, the C polynomial can be user specified as a design polynomial equivalent to the T polynomial in GPC. In either case, as shown in Section 7.3, ARM-MPC design does not require the solution of Diophantine equations and the noise/disturbance model is completely separated from the plant model in the prediction equation. The prediction equation is called a Separated Disturbance Prediction (SDP). This SDP enables independent on-line tuning of ARM-MPC for 'regulatory' control through simple algebraic manipulation and without re-calculation or re-design of the controller or solution of Diophantine equations. The ARM-MPC with SDP is shown in Figure 7.1.

7.2 Extended ARMarkov Model With Observer Polynomial

The extended ARMarkov model with a general disturbance model is defined as

$$\begin{aligned} y(k) &= -\sum_{j=1}^{n} \alpha_j y(k-\mu-j+1) + \sum_{j=0}^{\mu-1} h_j u(k-j) + \sum_{j=1}^{n} \beta_j u(k-\mu-j+1) \\ &\quad + \frac{C(z^{-1})}{\Delta} \epsilon(k) \qquad (7.1) \\ &= -\sum_{j=1}^{n} \alpha_j y(k-\mu-j+1) + \sum_{j=0}^{\mu} h_j u(k-j) + \sum_{j=2}^{n} \beta_j u(k-\mu-j+1) \\ &\quad + x(k) \qquad (7.2) \\ &= y_m(k) + x(k) \qquad (7.3) \end{aligned}$$

where C is a polynomial in z^{-1}, $x(k)$ is the disturbance or residual and $\Delta = 1 - z^{-1}$. Using a Prediction Error (PE) approach, both the plant and disturbance models can be determined by the following steps.

step 1: determine the model parameters assuming white noise disturbances

step 2: estimate the residual $x(k)$ and filter the residual to estimate the disturbance model

step 3: filter the input-output data by the disturbance model and estimate the model parameters

step 4: iterate the above steps until satisfactory convergence is obtained.

The $C(z^{-1})$ polynomial in the disturbance model in (7.1) can be treated as a design polynomial similar to the T-filter in GPC or it can be identified from process input/output data using the ARMarkov model structure as described above.

7.3 ARM-MPC With Observer Polynomial

The i step ahead output prediction equation using the model in (7.1) is written as

$$\begin{aligned} y(k+i) &= -\sum_{j=1}^{n} \alpha_j y(k-\mu-j+1+i) + \sum_{j=0}^{\mu-1} \frac{h_j}{\Delta} \Delta u(k-j+i) \\ &\quad + \sum_{j=1}^{n} \frac{\beta_j}{\Delta} \Delta u(k-\mu-j+1+i) \\ &\quad + \frac{C(z^{-1})}{\Delta} \epsilon(k+i) \end{aligned} \tag{7.4}$$

For the special structure of the ARMarkov model with $\mu \geq N_2$, the output terms on the RHS are available at time k. Therefore, the process and disturbance terms on the RHS need not be divided by a polynomial defining the process dynamics as in GPC (Clarke *et al.* 1987*b*) and the process and noise terms can be easily separated. The noise term is separated into past and future/present components using the Diophantine like expansion

$$\frac{C(z^{-1})}{\Delta} = E_i(z^{-1}) + z^{-i} \frac{F_i(z^{-1})}{\Delta} \tag{7.5}$$

where

$$E_i(z^{-1}) = 1 + e_1 z^{-1} + \cdots + e_{i-1} z^{-i+1} \tag{7.6}$$

$$F_i(z^{-1}) = f_0 + f_1 z^{-1} + \cdots + f_i z^{-i} \tag{7.7}$$

Since the disturbance model has only a differentiator, Δ in the denominator, there is no need to solve a Diophantine equation on-line. After some algebra, it can be shown that

$$\begin{aligned} e_i &= 1 + c_1 + \cdots + c_i \\ f_0 &= 1 + c_1 + \cdots + c_i, \; f_1 = c_{i+1}, \; \cdots, \; f_i = c_{2i-1} \end{aligned}$$

The minimum variance output prediction, obtained by putting the future noise component $E_i \epsilon(k+i)$ equal to zero, is

$$\begin{aligned} \hat{y}(k+i) &= -\sum_{j=1}^{n} \alpha_j y(k-\mu-j+1+i) + \sum_{j=0}^{\mu-1} \frac{h_j}{\Delta} \Delta u(k-j+i) \\ &\quad + \sum_{j=1}^{n} \frac{\beta_j}{\Delta} \Delta u(k-\mu-j+1+i) + \frac{F_i}{\Delta} \epsilon(k) \end{aligned} \tag{7.8}$$

Using (7.1)

$$\begin{aligned}\frac{\epsilon(k)}{\Delta} &= \frac{1}{C}[y(k) - (-\sum_{j=1}^{n}\alpha_j y(k-\mu-j+1) + \sum_{j=0}^{\mu-1} h_j u(k-j) \\ &\quad + \sum_{j=1}^{n}\beta_j u(k-\mu-j+1))] \qquad (7.9) \\ &= \frac{1}{C}x(k) = x^f(k) = \text{filtered residual at time } k\end{aligned}$$

Therefore, equation (7.8) can be written as

$$\begin{aligned}\hat{y}(k+i) &= -\sum_{j=1}^{n}\alpha_j y(k-\mu-j+1+i) + \sum_{j=0}^{\mu-1} s_j \Delta u(k-j+i) \\ &\quad + \sum_{j=1}^{n}\overline{\beta}_j \Delta u(k-\mu-j+1+i) + F_i x^f(k) \qquad (7.10)\end{aligned}$$

Equation (7.10) is referred to as a Separated Disturbance Predictor (SDP). The SDP has two parts. The first part consists of the first three terms which depend on the plant model. The second part, the last term in the SDP, depends only on the residual, x and the noise model and is completely independent of the plant model.

The prediction equation (7.10) can be written as

$$\hat{y}(k+i) = G_i(z^{-1})\Delta u(k+i-1) + \widehat{y}(k+i|k) \qquad (7.11)$$

where G_i contains the first i step response coefficients and the 'free response' is defined as

$$\begin{aligned}\widehat{y}(k+i|k) &= \sum_{j=i+1}^{\mu} s_j \Delta u(k-j+i) + \sum_{j=2}^{n}\overline{\beta}_j \Delta u(k-\mu-j+1+i) \\ &\quad - \sum_{j=1}^{n}\alpha_j y(k-\mu-j+1+i) + F_i x^f(k) \qquad (7.12) \\ &= z^{-i}\bar{S}\Delta u(k+i-1) + z^{-\mu}\bar{B}\Delta u(k+i-1) + z^{-(\mu-i)}\bar{A}y(k) \\ &\quad + F_i x^f(k) \qquad (7.13)\end{aligned}$$

where

$$\begin{aligned}\bar{S} &= s_{i+1} + s_{i+2}z^{-1} + \cdots + s_\mu z^{-(\mu-1-i)} \\ \bar{B} &= \overline{\beta}_2 + \overline{\beta}_3 z^{-1} + \cdots + \overline{\beta}_n z^{-(n-2)} \\ \bar{A} &= \alpha_1 + \alpha_2 z^{-1} + \cdots + \alpha_n z^{-(n-1)}\end{aligned}$$

The 'free response' includes the $F_i x^f(k)$ term to accommodate disturbances. A Model-based Predictive Controller (MPC) can therefore be designed following the same procedure as described in Chapter 6. The ARM-MPC using SDP is shown in Figure 7.1.

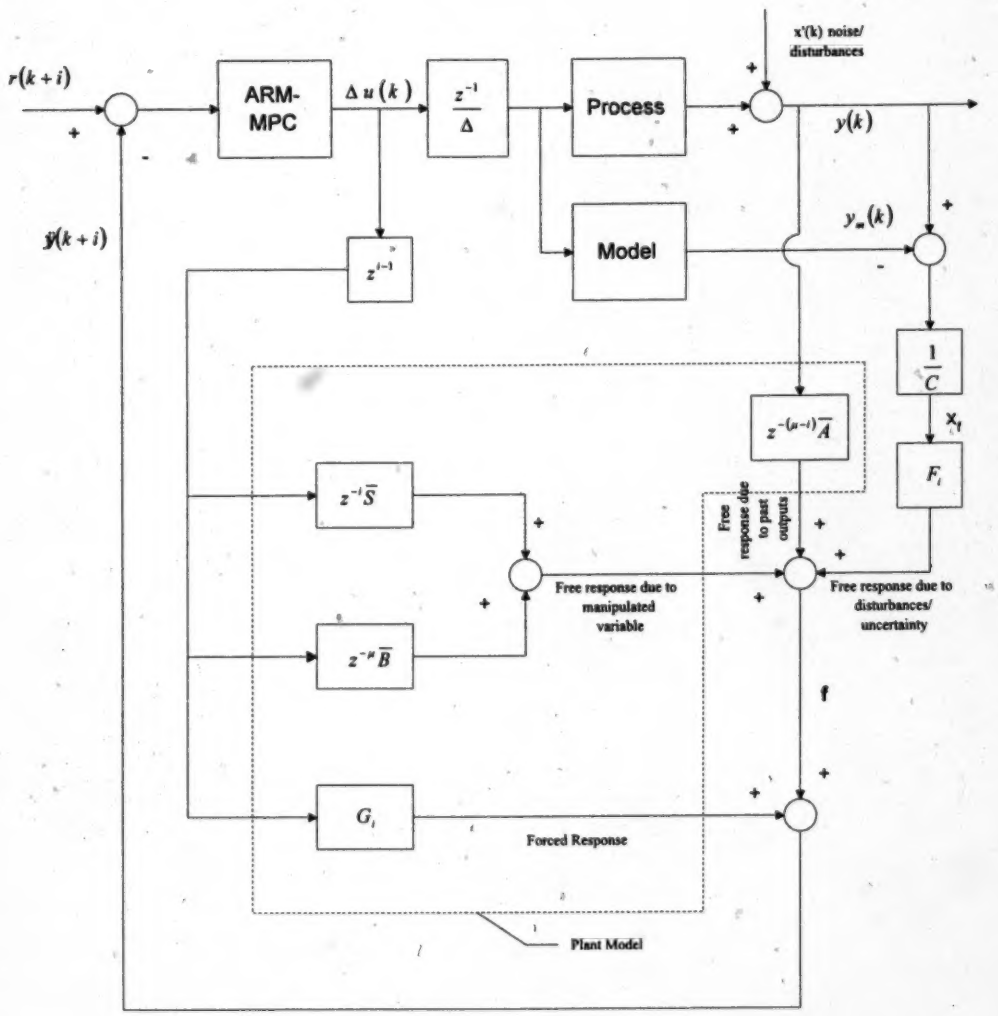


Figure 7.1: ARMarkov model-based predictive controller (ARM-MPC) using the separated disturbance predictor (SDP).

The 'free response' vector is defined as

$$f = [\widehat{y}(k+1|k), \widehat{y}(k+2|k), \cdots, \widehat{y}(k+N_2|k)]^T \tag{7.14}$$

and the future control vector as

$$\hat{\mathbf{u}} = [\Delta u(k), \Delta u(k+1), \cdots, \Delta u(k+M)]^T \tag{7.15}$$

The predicted output vector is defined as $\hat{\mathbf{Y}} = G\hat{\mathbf{u}} + f$. The elements of the '*dynamic matrix*', G are step-response coefficients. The unconstrained solution to the control problem is $\hat{\mathbf{u}} = \left(G^T Q G + \lambda R\right)^{-1} G^T Q \left(\mathbf{r} - f\right)$. *The advantage of this SDP formulation is that the servo and regulatory responses of the resulting MPC can be tuned independently because of the separation of the terms in (7.10).*

7.3.1 Flexibility of Disturbance Predictions

It is clear from equation (7.10) and Figure 7.1 that the disturbance prediction and feedback is completely independent of the process model. It can, of course, be tuned by adjusting C. However, even greater flexibility can be obtained by adjusting the disturbance prediction horizon, N_d, as illustrated by the following cases.

1. DMC type disturbance prediction is obtained by putting $C = 1$ in (7.1) as noted above. This is equivalent to making $N_d = 1$ and holding the prediction horizon constant for all future intervals.

2. GPC type disturbance prediction is obtained by using the same C in (7.1) as in GPC. This would imply putting $N_d = p$ (the output prediction horizon). However, any $1 \leq N_d \leq N_2$ can be used and the disturbance prediction for intervals more than N_d steps in the future either held constant (compare as DMC) or set to any other value, e.g. zero or decrease to zero over $N_2 - N_d$ intervals.

3. Different "filtering" and "prediction" methods can be used over different subintervals of $1 \leq N_d \leq N_2$.

This type of flexibility in the disturbance prediction permits "intuitive" algorithms, e.g. implemented by an expert system, as well as theoretical ones. Obvious practical advantages include:

a. if a large, unexpected and unexpandable step disturbance occurs then attenuate the prediction. If the step disturbance is confirmed by measurements over the following intervals then (rapidly) increase the prediction.

b. make N_d directly proportional to some measure of the reliability/certainty of the disturbance, e.g. be conservative and put $N_d = 1$ as in DMC or be aggressive and use $1 \leq N_d \leq N_2$ with an aggressive C as in GPC.

c. use historic disturbance data or disturbance patterns from files.

7.3.2 Limiting Case of ARM-MPC

ARM-MPC with Type-I Disturbances

In process control, it is very common to assume random step-type disturbances or Type-1 disturbances, which, in the simplest case, are modelled by $\epsilon(k)/\Delta$. For the ARMarkov model, this is equivalent to put $C = 1$ in (7.1).

With $C = 1$ in (7.1), $E_i = 1$ and $F_i = 1$ and the prediction equation is the same as (7.10) except that the last term simplifies to $\hat{\epsilon}(k)$.

7.4 Simulation Examples

The theory developed in the previous sections is illustrated through simulation examples in this section. ARMarkov identification along with an ARM-MPC is used to control the following third order process

$$G(z) = \frac{0.0077z^{-1} + 0.0212z^{-2} + 0.0036z^{-3}}{1 - 1.9031z^{-1} + 1.1514z^{-2} - 0.2158z^{-3}} \tag{7.16}$$

Example 7.1

Assume the process is subjected to step disturbances. The ARMarkov model order is assumed to be $n = 3$ and the number of Markov parameters in the model is $\mu = 10$. The controller is designed as in section 7.3. The tuning parameters for the controller design are as follows: $N_2 = 10$, $M = 1$, $Q = R = I$ and $\lambda = 0.0$. The disturbance model used in the ARM-MPC is $\hat{N} = \frac{1-0.8z^{-1}}{\Delta}$ i.e. $C = 1 - 0.8z^{-1}$ in (7.1). Both ARM-MPC and GPC are designed for this process with the same C polynomial and DMC is designed with $C = 1$. The results are shown in Figure 7.2. Comparison of Figures 7.2 and 7.3 shows that the aggressiveness of disturbance rejection is reduced significantly by the presence of the observer polynomial, C in both GPC & ARM-MPC. (DMC response remains the same with $C = 1$). Input variances and performance costs are compared for GPC & ARM-MPC in Table 7.1.

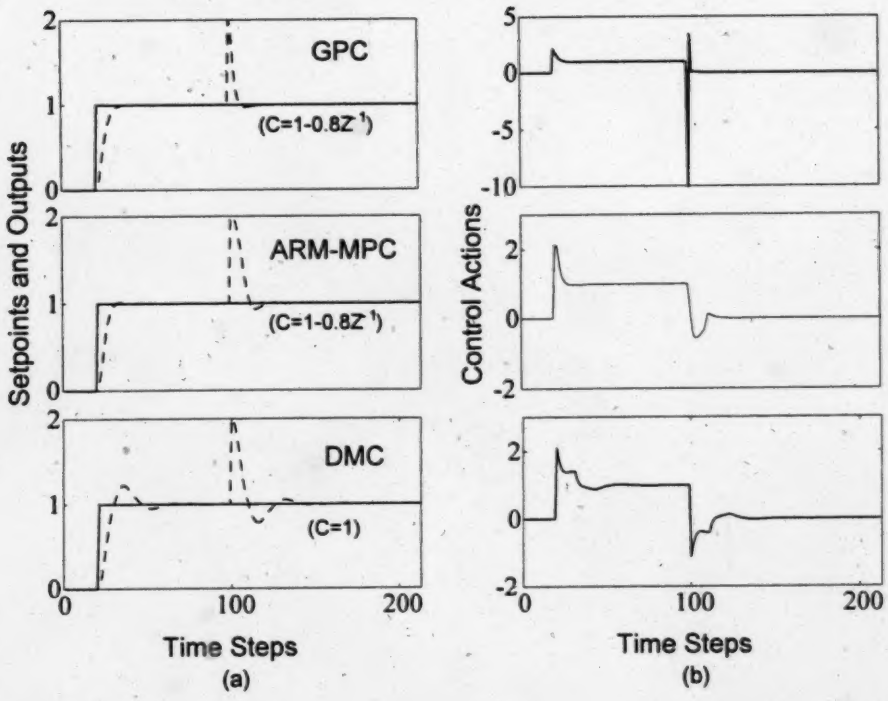


Figure 7.2: Comparison of responses using DMC, GPC and ARM-MPC with a Type-1 disturbance model in DMC and with an observer polynomial, $C = 1 - 0.8z^{-1}$ in GPC & ARM-MPC. In (a) 'solid'→ setpoints and 'dash'→ outputs. A unit step disturbance was added at time 100.

Both Figure 7.2 and Table 7.1 show that GPC is more aggressive than ARM-MPC which is expected. Although the performance cost in GPC is smaller, it requires larger control input moves. Moreover, as discussed in Section 7.3, *on-line tuning* of ARM-MPC using C does not require redesigning the controller or re-solving the Diophantine equations. This is a direct result of using the SDP.

Example 7.2

Now consider the same process in (7.16), but assume that the process is subjected to colored noise generated by passing white noise through the disturbance transfer function, $N = \frac{1-0.9z^{-1}}{1-1.9031z^{-1}+1.1514z^{-2}-0.2158z^{-3}}$. Process responses using GPC and ARM-MPC are shown in Figure 7.4. The controller parameters are the same as in (7.1). Input variances and performance costs are compared in Table 7.2. Results similar to

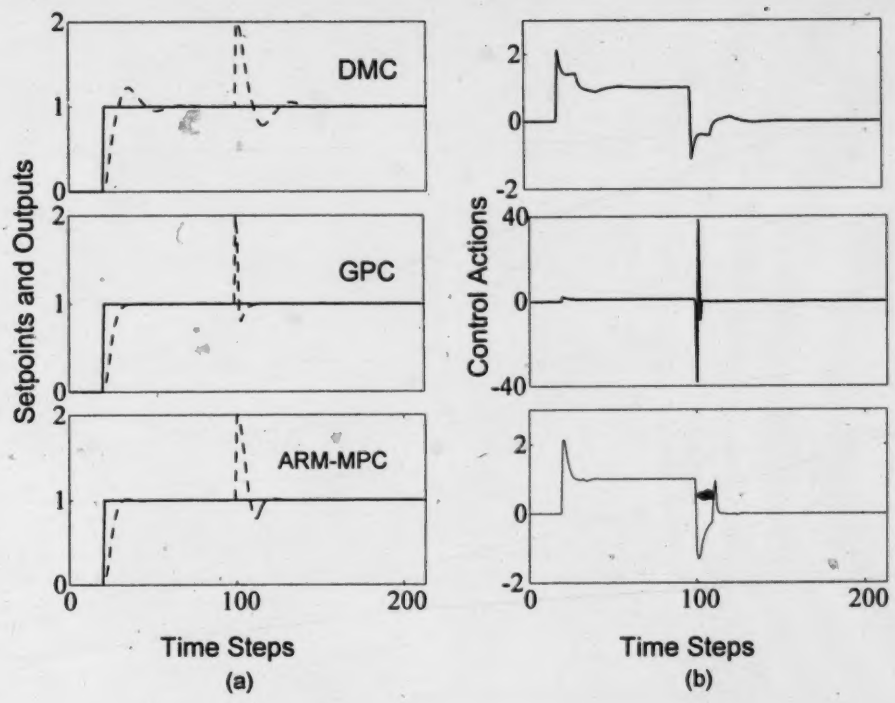


Figure 7.3: Comparison of responses from DMC, GPC and ARM-MPC with a Type-1 disturbance model. In (a) 'solid'→ setpoints and 'dash'→ outputs. A unit step disturbance was added at time 100.

Example 7.1 are obtained. The *outputs* plotted in Figure 7.4 do not show the filtering effect as strongly as the control variable because the noise is added directly to the output.

In the above analysis, SISO systems are considered for simplicity. However, ARM-MPC can also be applied to MIMO systems by using the block parameters estimated using the ARMarkov method for MIMO systems as extended by Kamrunnahar *et al.* (2000) and described in Chapter 2.

7.5 Conclusions

- The disturbance feedback and prediction (observer) are independent of the plant model. The proposed ARM-MPC controller can be tuned independently for servo and regulatory responses because of the Separated Disturbance Predictor

Table 7.1: Input Variances and Performance Costs for Different Controllers with $C = 1 - 0.8z^{-1}$

	GPC	ARM-MPC
var(u)	0.840	0.295
Performance Cost, J	5.84	9.03

Table 7.2: Input Variances and Performance Costs for Different Controllers with $C = 1 - 0.8z^{-1}$

	GPC	ARM-MPC
var(u)	0.196	0.106
Performance Cost, J	3.820	3.924

(SDP) developed as part of this thesis.

- As in GPC, the observer polynomial, C can be specified by the user to modify the system response to disturbances e.g. to make the response slower or more conservative, but with ARM-MPC on-line tuning is possible without solving the Diophantine equations at every sampling instant.

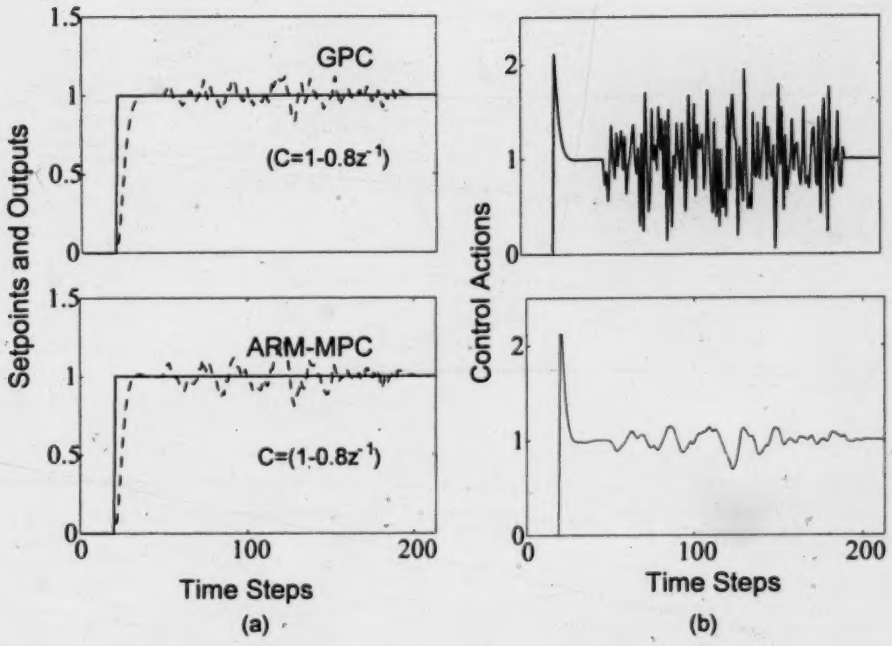


Figure 7.4: Responses from an ARMAX process using GPC and ARM-MPC with an observer polynomial, $C = 1 - 0.8z^{-1}$. In (a) 'solid'→ setpoints and 'dash'→ outputs.

Chapter 8

Dual-Model State-space ARM-MPC

8.1 Introduction

Linear, time invariant (LTI) models typically used in the design of model-based predictive controllers (MPC) include: transfer function, state space, Finite Step Response (FSR) and Finite Impulse Response (FIR). The transfer function and state space models are equivalent in the sense that each can be transformed into the other forms. They can also be transformed into FSR or FIR models. The type of model to be used in MPC generally depends on the application of the controller *e.g.* for adaptive control, designs based on transfer function models (*e.g.* GPC) are more common whereas for complex, large scale systems with several inputs and outputs, state space models are more convenient. Ricker (1991)discussed the design of MPC using general state space models. Meadows *et al.* (1995*b*) discussed implementation of MPC in the state space. The main advantages of using state space models are the availability of well-developed and established state space theory for the analysis of the MPC systems as well as the convenience of application to large scale MIMO systems. On the other hand, the drawback of using state space models is that the states in the model may not have any physical meanings to the plant operators and may require state estimation algorithms rather than simple direct measurement.

FIR and FSR models are the simplest models used in MPC design. Navratil *et al.* (1988*a*)and Li *et al.* (1989)have shown that MPC's based on step response models can be rearranged into a state space form and then the state space MPC can be analyzed

for stability and other properties using the well-known state space theorems. The advantage is two-fold: use of a simple step response model and at the same time state space theorems can be used for analysis. As pointed out by Ricker (1991), FSR or FIR models can be viewed as a high order state space model by considering the past inputs (or input deviations) as the states. Morari and Lee (1991)formulated an MPC using this concept and extended it to processes with pure integrators. Lee *et al.* (1994) developed a state-space model in terms of step-response coefficients for systems with stable and/or integrating dynamics and extended the conventional MPC to handle stochastic and white noise disturbances without solving a large-order Riccati equation. Ruscio (1997*a*) presented an extended state-space MPC formulation and showed that both general linear state-space models and impulse response models fit into that framework.

Qi and Fisher (1993) and Qi (1997)introduced a Dual-Model Predictive Controller (DMPC) design that uses step response coefficients for the fast dynamics and a low order parametric model for the slow or unstable dynamics of the process. The state space form of the dual model MPC has a special (simplified) structure that offers several advantages. For example, Qi and Fisher (1994) showed that use of the special structure, dual-model MPC lead to less conservative robustness bounds than conventional state space models. Another advantage of the step/impulse response type models in the state space interpretation of MPC's is that the states of the model have physical meaning *i.e.* the states are the predicted future outputs of the plant.

The ARM-MPC described in Chapter 6 of this thesis used an input-output model that is a combination of parametric and non-parametric models. For convenience of analysis of the ARM-MPC system using the well-developed state space theorems available in the literature, an equivalent state-space formulation of the ARM-MPC is developed in this chapter. The input-output ARM-MPC and the state-space ARM-MPC are equivalent. However, the special dual-model state space structure leads to some useful MPC properties *e.g.* improved robustness bounds as discussed in Chapter 10. The state space formulation of the ARM-MPC is similar to the Dual-Model Predictive Controller (DMPC) of Qi (1997) except that the ARM-MPC needs a smaller number of Markov parameters and a smaller prediction horizon because of the full input-output model structures.

An overview of the state space formulation of MPC using step response models is presented in Section 8.2. In Section 8.3, a state space ARM-MPC is developed starting from an input-output ARMarkov model. A recursive control move calculation that avoids on-line redesign of the controller is discussed in Section 8.4. Application of the steady state error weighting and its implications for the ARM-MPC are described in Section 8.5. Simulation examples illustrating the implementation of ARM-MPC are given in Section 8.6.

Note: In this chapter a state space model is derived directly from the ARMarkov input-output model and formulated with a special "dual-model structure" that is shown to offer significant advantages, e.g. in robustness analysis. The model is a subset of the classical $[A, B, C, D]$ state space model derived in the next chapter.

8.2 An Overview of the State-space Formulation of MPC

8.2.1 Model Formulation

Assume that a SISO process can be represented by an FIR or equivalently by an FSR model. This model can be formulated in a standard state space form by defining the predicted output trajectory as the state variables as (Li *et al.* 1989, Morari and Lee 1991, Lee *et al.* 1994, Qi and Fisher 1993):

$$\begin{aligned} X(k) &= \Phi X(k-1) + \theta \Delta u(k-1) \\ y(k) &= HX(k) \end{aligned} \tag{8.1}$$

where

$$X(k) = [y_m^*(k|k), y_m^*(k+1|k) \cdots y_m^*(k+n|k)]^T_{(n1+1)\times 1} \tag{8.2}$$

$$X(k-1) = [y_m^*(k-1|k-1), y_m^*(k|k-1) \cdots y_m^*(k+n1-1|k-1)]^T_{(n1+1)\times 1} \tag{8.3}$$

$$\Phi = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & r_{n_r} & \cdots & r_2 & r_1 \end{bmatrix}_{(n1+1)\times(n1+1)} \tag{8.4}$$

$$\theta = \begin{bmatrix} s_1 & s_2 & s_3 & \cdots & s_{n1+1} \end{bmatrix}^T \tag{8.5}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}^T \tag{8.6}$$

$y_m^*(*|k)$ represents the contribution of all past control actions to the future output trajectory, k represents the current sample time, $\{s_j,\ j = 1, 2, \ldots, n1+1\}$ are the discrete step response coefficients. The coefficients$\{r_j,\ j = 1, 2, \ldots, n_r\}$ are defined by different researchers in different ways. *e.g.* for a full step response model (used in DMC), as defined by Li *et al.* (1989)for a stable process,

$$r_1 = 1,\ n_r = 1$$

Morari and Lee (1991)defined r_j for integrating systems as

$$r_1 = 2,\ r_2 = -1,\ n_r = 2$$

and Qi (1997)represented r_j by the coefficients of a parametric model $A_2\Delta$ that represents the slow or unstable dynamics of the process step response persisting after the initial $\{s_1, s_2, \ldots, s_{n1+1}\}$ points and generalizes the above two definitions as special cases of

$$A_2\Delta = 1 - r_1 z^{-1} - \cdots - r_{n_r} z^{-n_r}$$

The above formulation is valid for MIMO systems using the vector forms of the inputs-outputs as described in Morari and Lee (1991).

8.2.2 State Estimation

Using the measurements at time k, an optimal state estimator can be designed using classical observer theory. For example, the two stage (Kalman filter type) state estimator form (Navratil *et al.* 1988*b*, Lee *et al.* 1994), can be written as

$$\begin{aligned} \hat{X}(k) &= \Phi X^*(k-1) + \theta \Delta u(k-1) \\ \hat{y}(k) &= H\hat{X}(k) \\ X^*(k) &= \hat{X}(k) + K\left(y(k) - \hat{y}(k)\right) \end{aligned} \tag{8.7}$$

where $X^*(k)$ is the estimated state variable vector, $y(k)$ is the actual output measurement at time k and K is the generalized feedback estimator gain which can be calculated by using the classical solution to the Riccati equation.

8.2.3 Output Prediction

The estimated states consist of the predicted output components due to past control action. These output components are known collectively as the 'free response' which is defined as

$$Y_m(k+i|k) = \Phi_{N_2} X^*(k) \tag{8.8}$$

where N_2 is the prediction horizon and

$$\Phi_{N_2} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \cdots & 0 \end{bmatrix}_{N_2 \times (n1+1)} \tag{8.9}$$

The output prediction due to both past and future control moves is written as

$$\widehat{Y}(k+i|k) = Y_m(k+i|k) + G\widehat{\mathbf{u}} \tag{8.10}$$

where

$$G = \begin{bmatrix} s_1 & 0 & 0 & \cdots & 0 \\ s_2 & s_1 & 0 & \cdots & 0 \\ s_3 & s_2 & s_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_M & s_{M-1} & s_{M-2} & \cdots & s_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{N_2} & s_{N_2-1} & s_{N_2-2} & \cdots & s_{N_2-M+1} \end{bmatrix}_{N_2 \times M} \tag{8.11}$$

is the well known 'dynamic matrix' present in all MPC designs, M is the control horizon and $\widehat{\mathbf{u}}$ represents the present/future control vector

$$\widehat{\mathbf{u}} = [\Delta u(k), \Delta u(k+1), \cdots, \Delta u(k+M-1)]^T$$

8.2.4 Control Move Calculation

The unconstrained MPC solution can be obtained by calculating the control trajectory, $\widehat{\mathbf{u}}$, that minimizes a performance criterion

$$J = E^T Q E + \lambda \widehat{\mathbf{u}}^T R \widehat{\mathbf{u}} \tag{8.12}$$

where Q and λR are the output and input weightings respectively, E is the error between the predicted output and future set-point or target vector, $Y_{sp} = [y_{sp}(k+1), y_{sp}(k+2), \cdots, y_{sp}(k+N_2)]^T$ i.e.

$$\begin{aligned} E &= Y_{sp} - Y_m(k+i|k) \\ &= Y_{sp} - \Phi_{N_2} X^*(k) \end{aligned} \tag{8.13}$$

Minimization of the performance index in (8.12) leads to the following least-squares control problem

$$\left(G^T Q G + \lambda R\right) \hat{\mathbf{u}} = G^T E \tag{8.14}$$

The solution to the control problem is

$$\hat{\mathbf{u}} = A^* E \tag{8.15}$$

where the $M \times N_2$ pseudo-inverse matrix, A^* is written as

$$A^* = (G^T Q G + \lambda R)^{-1} G^T Q \tag{8.16}$$

8.2.5 Closed-loop Formulation

In accordance with the widely used receding horizon principle, only the first control element, $\Delta u(k)$, of the control trajectory, $\hat{\mathbf{u}}(k)$, is implemented at time k and the control law is executed at every sampling instant

$$\begin{aligned} \Delta u(k) &= C^T \hat{\mathbf{u}}(k) \\ &= C^T A^* \left[Y_{sp} - \Phi_{N_2} X^*(k)\right] && (8.17) \\ &= -K_{mpc} X^*(k) + R_{mpc} Y_{sp} && (8.18) \end{aligned}$$

where $C^T = [\,1,\ 0,\ \cdots,\ 0\,]$. This is a state feedback controller and, in general, the feedback gain is time invariant but is a function of the MPC tuning parameters i.e.

$$\begin{aligned} K_{mpc} &= \mathcal{F}_1(\lambda, N_2, M) = C^T A^* \Phi_{N_2} \\ R_{mpc} &= \mathcal{F}_2(\lambda, N_2, M) = C^T A^* \end{aligned} \tag{8.19}$$

Using the model in (8.1) for the open-loop process, the closed-loop formulation can be written as

$$\begin{aligned} X(k+1) &= (\Phi - \theta K_{mpc}) X(k) + \theta R_{mpc} Y_{sp}(k) \\ y(k) &= HX(k) \end{aligned} \tag{8.20}$$

After introducing the state error vector,

$$\overline{X}(k) = X(k) - X^{*}(k) \tag{8.21}$$

and combining equations (8.1), (8.7) and (8.21), the following equation is obtained for the feedback observer

$$\overline{X}(k) = (I - KH)\Phi\overline{X}(k-1) \tag{8.22}$$

8.2.6 Closed-loop Properties

The performance of the closed-loop system can be determined by theoretical analysis and/or simulation based on the above equations. Later sections will make use of the fact that:

- For an asymptotically stable closed-loop system, all eigenvalues of the closed-loop system matrix, $\Phi - \theta K_{mpc}$ and the observer matrix, $(I - KH)\Phi$ must be within the unit circle.

8.3 State Space ARM-MPC based on the Input-output ARMarkov Model

Rewrite the input-output extended ARMarkov model in (6.3) with a type-1 disturbance model as

$$\begin{aligned} y(k) &= \sum_{j=1}^{\mu-1} h_j u(k-j) + \sum_{j=1}^{n} \beta_j u(k-\mu-j+1) - \sum_{j=1}^{n} \alpha_j y(k-\mu-j+1) \\ &\quad + \frac{1}{\Delta}\varepsilon(k) \qquad (8.23) \\ &= \sum_{j=1}^{\mu-1} h_j u(k-j) + Xm + \frac{1}{\Delta}\varepsilon(k) \qquad (8.24) \end{aligned}$$

where

$$Xm = \sum_{j=1}^{n} \beta_j u(k-\mu-j+1) - \sum_{j=1}^{n} \alpha_j y(k-\mu-j+1)$$

$$\begin{aligned} y(k) &= y(k-1) + \sum_{j=1}^{\mu-1} \Delta h_j u(k-j) + \sum_{j=1}^{n} \Delta\beta_j u(k-\mu-j+1) \\ &\quad - \sum_{j=1}^{n} \Delta\alpha_j y(k-\mu-j+1) + \varepsilon(k) \end{aligned} \tag{8.25}$$

As was done in the definition of the states in (8.2), the 'free responses' are defined as the states of the ARMarkov model in (8.23) as

$$\tilde{X}(k) = [y_m^*(k|k), y_m^*(k+1|k) \cdots y_m^*(k+\mu|k)]^T_{(\mu+1)\times 1} \tag{8.26}$$

$$\tilde{X}(k-1) = [y_m^*(k-1|k-1), y_m^*(k|k-1) \cdots y_m^*(k+\mu-1|k-1)]^T \tag{8.27}$$

where $(\mu+1)$ is the number of predicted outputs.

After some algebraic manipulation of equation (8.25), it can be shown that

$$y_m^*(k+i|k) = y_m^*(k+i|k-1) + s_{i+1}\Delta u(k-1) \quad \text{for } i = 0, \cdots (\mu-1) \tag{8.28}$$

$$\begin{aligned} y_m^*(k+\mu|k) &= y_m^*(k+\mu-1|k-1) + s_\mu \Delta u(k-1) + \beta_2 \Delta u(k-1) + \\ &\quad \cdots + \beta_n \Delta u(k-n+1) - \alpha_1 y(k) \\ &\quad - \sum_{j=2}^{n} (\alpha_j - \alpha_{j-1}) y(k-j+1) + \alpha_n y(k-n) \end{aligned} \tag{8.29}$$

for $i = \mu$. The state space form of the ARM-MPC prediction model is, then, written in the same form as (8.1)

$$\begin{aligned} \hat{X}_{arm}(k) &= \Phi_{arm}\hat{X}_{arm}(k-1) + \theta_{arm}\Delta u(k-1) \\ \hat{y}(k) &= H_{arm}\hat{X}_{arm}(k) \end{aligned} \tag{8.30}$$

where

$$H_{arm} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}^T_{1\times(\mu+n)} \tag{8.31}$$

$$\Phi_{arm} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix}_{(\mu+n)\times(\mu+n)}, \quad \theta_{arm} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \tag{8.32}$$

$$\phi_{11} = \left[\begin{array}{cc} 0_{\mu\times 1} & I_{\mu\times\mu} \end{array}\right]_{\mu\times(\mu+1)}, \quad \phi_{12} = 0_{\mu\times(n-1)}$$
$$S_1 = \left[\begin{array}{cccc} s_1 & s_2 & \cdots & s_\mu \end{array}\right]^T_{\mu\times 1}$$

The structures of S_2, ϕ_{21} and ϕ_{22} vary with the order of the residual parametric model used to represent the slow dynamics of the process. After the initial $(\mu - 1)$ impulse response points in the ARMarkov model, $(n-1)$ more states are added to the state vector in (8.26) to represent the dynamics corresponding to X_m in the ARMarkov model in (8.24). A low order is used to model the slow dynamics, X_m and, in this work is restricted to be $1 \leq n \leq 3$. For example, using $n = 3$ leads to the following additional states

$$\hat{X}_{ar} = \left[\begin{array}{c} x_{ar1} \\ x_{ar2} \end{array}\right] = \left[\begin{array}{c} y(k-1) \\ \alpha_3 y(k-2) + \beta_3 \Delta u(k-1) \end{array}\right] \tag{8.33}$$

The resulting state vector and $S_2, \phi_{21}, \phi_{22}$ matrices can be written as

$$\hat{X}_{arm}(k) = \left[\begin{array}{c} \tilde{X}(k) \\ x_{ar1} \\ x_{ar2} \end{array}\right] \tag{8.34}$$

$$\phi_{21} = \left[\begin{array}{cccc} -(\alpha_2 - \alpha_1) & -\alpha_1 & 0_{1\times(\mu-3)} & 1 \\ 1 & 0 & 0_{1\times(\mu-3)} & 0 \\ 0 & \alpha_2 & 0_{1\times(\mu-3)} & 0 \end{array}\right]^T_{n\times(\mu+1)} \tag{8.35}$$

$$\phi_{22} = \left[\begin{array}{cc} -(\alpha_3 - \alpha_2) & 1 \\ 0 & 0 \\ \alpha_3 & 0 \end{array}\right]^T_{n\times(n-1)} \tag{8.36}$$

and
$$S_2 = \left[\begin{array}{ccc} s_{\mu+1} - \alpha_1 s_1 & 0 & \beta_3 \end{array}\right]^T \tag{8.37}$$

Model orders $n < 3$ simplify the formulation further but are not discussed here due to their simplicity.

State estimation, output prediction and control move calculation follow the same principles as discussed in subsections 8.2.2 through 8.2.4 with Φ_{N_2} a $(N_2 \times (\mu + n))$ matrix and other matrices of appropriate dimensions.

8.3.1 Feedback Observer Design

Introducing process and measurement noise, the process model (8.30) can be written as

$$\begin{aligned} \hat{X}_{arm}(k) &= \Phi_{arm}\hat{X}(k-1) + \theta_{arm}\Delta u(k-1) + \Gamma_w \Delta w(k) \\ \hat{y}(k) &= H_{arm}\hat{X}_{arm}(k) + \nu(k) \end{aligned} \tag{8.38}$$

Following the same procedure in subsection 8.2.2, an optimal, Kalman filter type, two stage observer for the state space ARM-MPC can be designed as

$$\begin{aligned} \tilde{X}_{arm}(k) &= \Phi_{arm}X^*_{arm}(k-1) + \theta\Delta u(k-1) \\ \hat{y}(k) &= H_{arm}\tilde{X}_{arm}(k) \\ X^*_{arm}(k) &= \tilde{X}_{arm}(k) + K_{arm}(y(k) - \hat{y}(k)) \end{aligned} \tag{8.39}$$

where $X^*_{arm}(k)$ is the estimated state vector and K_{arm} is the feedback observer (Kalman filter) gain. Assume that $\Delta w(k), \nu(k)$ are white noise. Then the observer gain, K_{arm} can be calculated as

$$K_{arm} = \Phi_{arm}PH^T_{arm}\left(R_2 + H_{arm}PH^T_{arm}\right)^{-1}$$

where P is the steady state solution of the Riccati equation

$$\begin{aligned} P(k+1) &= \Phi_{arm}P(k)\Phi^T_{arm} + R_1 - \Phi_{arm}P(k)H^T_{arm} \\ &\quad \left(R_2 + H_{arm}P(k)H^T_{arm}\right)^{-1} H_{arm}P(k)\Phi^T_{arm} \end{aligned}$$

and R_1, R_2 are the covariance matrices for $w(k)$ and $\nu(k)$ respectively and are usually user-specified.

Assuming Γ_w defines a step-type disturbance model and $\Delta w(k)$ is white noise, the solution of the Riccati equation can be simplified as described in Morari and Lee (1991).

Observer Design Using Pole Placement

Qi and Fisher (1993)developed a state observer design for the *dual-model* MPC using pole placement and presented a simplified solution for the calculation of the dead-beat

observer gain. Using a similar pole placement approach, a simplified observer can be designed for the state space ARM-MPC as discussed below.

Define the state error vector as

$$\bar{X}(k) = \tilde{X}_{arm}(k) - X^*_{arm}(k) \tag{8.40}$$

Combining (8.30), (8.39) and (8.40), the closed-loop observer equation is written as

$$\bar{X}(k) = (I - K_{arm}H_{arm})\,\Phi_{arm}\bar{X}(k-1)$$

For the observer to be asymptotically stable and convergent, it must satisfy the condition

$$\lim_{k\to\infty} \bar{X}(k) = 0$$

and the eigenvalues of the observer must be within the unit circle. The characteristic equation of the observer is expressed as

$$\det\left[\lambda I - (I - K_{arm}H_{arm})\,\Phi_{arm}\right] = 0 \tag{8.41}$$

The poles *i.e.* the eigenvalues of the observer can be assigned as per desired observer dynamics and the observer gain K_{arm} can be calculated assuming

$$K_{arm} = \begin{bmatrix} k_1 & k_2 & \cdots & k_{\mu+n} \end{bmatrix}^T$$

A dead-beat observer that gives fast state convergence is designed by placing all the eigenvalues at the origin and is common in control literature. Due to the structure of the state space system matrix, Φ_{arm} with zeros and ones in ϕ_{11}, for an open-loop stable, SISO process, the dead-beat observer gain for the ARM-MPC simplifies to

$$K_{arm} = \left[\underbrace{1, 1, \cdots, 1}, 1+\alpha_1, 0, \cdots, 0\right]^T$$

where $1+\alpha_1$ is at the $(1,\mu)\,th$ position.

8.3.2 Closed-loop Formulation

The unconstrained ARM-MPC solution can be obtained by using the same performance criterion and following the same procedure as in subsection 8.2.4. The calculated current control move is

$$\Delta u(k) = C^T \hat{\mathbf{u}}(k) \tag{8.42}$$

$$= -K_{arm-mpc}X^*_{arm}(k) + R_{arm-mpc}Y_{sp} \tag{8.43}$$

The closed-loop formulation of the ARM-MPC system can be expressed as

$$\begin{aligned}\hat{X}_{arm}(k+1) &= (\Phi_{arm} - \theta_{arm}K_{arm-mpc})\hat{X}_{arm}(k) + \theta_{arm}R_{arm-mpc}Y_{sp}(k) \quad (8.44)\\ \hat{y}(k) &= H_{arm}\hat{X}_{arm}(k)\\ \overline{X}_{arm}(k) &= (I - K_{arm}H_{arm})\Phi_{arm}\overline{X}_{arm}(k-1)\end{aligned}$$

8.4 Recursive Control Move Calculation for Increasing Control Horizon

The control horizon, M is a very important and effective tuning parameter in MPC. Increasing/decreasing the control horizon usually requires that the MPC controller be redesigned on-line which demands significant computational effort. Qi (1997)introduced a recursive calculation for increasing the control horizon for SISO systems and used it for dynamic tuning of Dual Model Predictive Control (DMPC). This recursive calculation of control moves is applicable to any MPC design and does not require redesign of the controller. This saves computation which includes the advantage of on-line inversion of large matrices.

In this chapter, the recursive control move calculation is applied to the state space ARM-MPC. It is also shown that this concept can be extended to MIMO systems by simply using step response block coefficients and changing some matrix dimensions.

For simplicity, assume that the output and control weightings are $Q = I$ and $\lambda R = \lambda I$ respectively. Then the MPC objective function in (8.12) becomes

$$J = E^T E + \lambda \hat{\mathbf{u}}^T \hat{\mathbf{u}}$$

and minimization of this performance criteria leads to the least-squares control problem

$$(G^T G + \lambda I)\,\hat{\mathbf{u}} = G^T E$$

For the control horizon, $M = m$, denote the dynamic matrix as G_m and the present /future control move vector as $\hat{\mathbf{u}}_m^0$. Assume that an extra control term Δu_{m+1} is added to the MPC designed with a control horizon $M = m$. The new dynamic matrix G_{m+1}

can be expressed in terms of the previous matrix G_m and a new coefficient vector x_{m+1} as:

$$G_{m+1} = [G_m,\ x_{m+1}] \tag{8.45}$$

where

$$x_{m+1} = [0, \cdots, 0, s_1, \cdots s_{N_2-m}]^T_{(N_2 \times 1)} \tag{8.46}$$

such that

$$\begin{aligned} G_{m+1}^T G_{m+1} &= \begin{bmatrix} G_m^T \\ x_{m+1}^T \end{bmatrix} \begin{bmatrix} G_m & x_{m+1} \end{bmatrix} \\ &= \begin{bmatrix} G_m^T G_m & G_m^T x_{m+1} \\ x_{m+1}^T G_m & x_{m+1}^T x_{m+1} \end{bmatrix} \end{aligned}$$

$$G_{m+1}^T G_{m+1} + \Lambda = \begin{bmatrix} G_m^T G_m + \lambda I & G_m^T x_{m+1} \\ x_{m+1}^T G_m & x_{m+1}^T x_{m+1} + \lambda_{m+1} \end{bmatrix} \tag{8.47}$$

The least-squares solution to the future control problem can, then, be written as

$$\begin{bmatrix} G_m^T G_m + \lambda I & G_m^T x_{m+1} \\ x_{m+1}^T G_m & x_{m+1}^T x_{m+1} + \lambda_{m+1} \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{u}}_m^1 \\ \Delta u_{m+1} \end{bmatrix} = \begin{bmatrix} G_m^T \\ x_{m+1}^T \end{bmatrix} E \tag{8.48}$$

Lemma 8.1 *(Qi 1997, Qi* et al. *2001)The control vector, $\widehat{\mathbf{u}}_m^1$ containing $m+1$ future control moves (in terms of the $m-$vector solution, $\widehat{\mathbf{u}}_m^0$) is*

$$\widehat{\mathbf{u}}_m^1 = \widehat{\mathbf{u}}_m^0 - G_1 x_{m+1}^T \left(E - G_m \widehat{\mathbf{u}}_m^0\right) \tag{8.49}$$

$$\Delta u_{m+1} = G_2 x_{m+1}^T \left(E - G_m \widehat{\mathbf{u}}_m^0\right) \tag{8.50}$$

$$\widehat{\mathbf{u}}_m^0 = \left(G_m^T G_m + \lambda I\right)^{-1} G_m^T E$$

$$G_1 = G_3 G_2$$

$$G_2 = \left(x_{m+1}^T x_{m+1} + \lambda_{m+1} - x_{m+1}^T G_m G_3\right)^{-1}$$

$$G_3 = \left(G_m^T G_m + \lambda I\right)^{-1} G_m^T x_{m+1}$$

Proof. Assume a matrix

$$\Gamma = \begin{bmatrix} A & B \\ B^T & D \end{bmatrix} \tag{8.51}$$

and

$$\Gamma^{-1} = \psi = \begin{bmatrix} \psi_{11} & \psi_{12} \\ \psi_{21} & \psi_{22} \end{bmatrix} \tag{8.52}$$

Using the well known matrix inversion Lemma

$$\begin{aligned} \psi_{11} &= A^{-1} + A^{-1}B\left(D - B^TA^{-1}B\right)^{-1}B^TA^{-1} \\ \psi_{12} &= -A^{-1}B\left(D - B^TA^{-1}B\right)^{-1} \\ \psi_{21} &= -\left(D - B^TA^{-1}B\right)^{-1}B^TA^{-1} \\ \psi_{22} &= \left(D - B^TA^{-1}B\right)^{-1} \end{aligned}$$

Define

$$\begin{aligned} A &= G_m^TG_m + \lambda I \\ B &= G_m^Tx_{m+1} \\ D &= x_{m+1}^Tx_{m+1} + \lambda_{m+1} \end{aligned}$$

■

The proof follows from the substitution of the above definitions in (8.48). The proof is easily extended to the case where the control horizon varies from $M = m$ to $M = m + n$. This is discussed in Chapter 11.

Note that the term $(E - G_m\hat{\mathbf{u}}_m^0)$ occurs in both equations (8.49) and (8.49). Define $(E - G_m\hat{\mathbf{u}}_m^0)$ as the residual after implementation of m future control actions calculated using $M = m$ to minimize the prediction error, E. The above expression can be interpreted as described in the following remarks:

- *With an extra element in the control horizon, the new future control vector, $\hat{\mathbf{u}}_m^1$ with $(M = m + 1)$, can be obtained by using the original $(M = m)$ control vector $\hat{\mathbf{u}}_m^0$ plus a modification term as in (8.49).*

- *The effectiveness of increasing the control horizon can be evaluated by analyzing the residual term $(E - G_m\hat{\mathbf{u}}_m^0)$. From the point of view of system optimization, a large residual suggests that the control horizon should be increased until the residual becomes sufficiently small.*

- *The recursive update of the future control vector involves only the inversion of a lower dimensional* (n by n) *matrix,* G_2 *instead of an* $(m+1)$ *by* $(m+1)$ *matrix. For SISO systems, if only one control horizon is retuned from* m *to* $m+1, G_2$ *is scalar.*

Recursive Calculation for MIMO Systems

The recursive control calculation discussed above is easily extendable to MIMO systems. For MIMO systems, x_{m+1} in (8.46) is a $(l*(N_2+1)\times r)$ matrix instead of a vector and can be written as

$$x_{m+1} = \left[0, \cdots, 0, S_1^T, \cdots S_{p-m}^T\right]^T_{l(N_2+1)\times r}$$

where l and r are the number of outputs and inputs of the MIMO system. λ_{m+1} in (8.47) is a $(r \times r)$ matrix instead of a scalar. The dimensions of the other elements change accordingly without any difficulty. Lemma 8.1 and its proof are valid for both SISO and MIMO systems because only the dimensions of the matrices A, B, C, D change for MIMO systems without changing the matrix properties. D in (8.51) is a $(r \times r)$ matrix instead of a scalar as in SISO systems.

The remarks made in Section (8.2.3) also hold for MIMO systems with appropriate allowance for the matrix dimensions. The residual $(E - G_m\hat{\mathbf{u}}_m^0)$ is still a scalar but the recursive update requires the inversion of a $(r \times r)$ matrix, G_2.

Application of the recursive control calculations described in this section was illustrated by Kamrunnahar *et al.* (1998)for a (2×2) MIMO systems using a binary distillation column example.

8.5 Steady-state Weighting, γ_∞

Meadows and Rawlings (1993) and K.R. *et al.* (1994) introduced an infinite horizon in MPC to guarantee the stability of the feedback system. It was shown by researchers (Kwok and Shah 1994, Kwok 1992, Saudagar 1995) that steady-state weighting gives the effect of using an infinite prediction horizon without increasing the dimension of the problem significantly. It was also shown (Kwok and Shah 1994, Banerjee 1996) that it increases the robust stability of predictive controllers. Kwok and Shah (1994)

modified the Long-Range Predictive Control (LRPC) objective function by augmenting a steady-state weighting and showed that the steady-state weighting, γ_∞ gives the effect of a large prediction horizon without actually using one. Kwok (1992) showed that the γ_∞ parameter gives better stability properties than the move suppression factor λ. Saudagar (1995) presented a simplified form of the steady-state error weighting problem. Banerjee (1996) analyzed the robustness of GPC and Markov-Laguerre Model-based Predictive Controllers using steady-state error weighting. In this chapter, the effect of γ_∞ on the state-space ARM-MPC is investigated.

8.5.1 Steady-state Output Prediction

Rewrite the state-space model of the process in (8.1) as

$$\begin{aligned} \hat{X}(k+1) &= \Phi\hat{X}(k) + \theta\Delta u(k) \\ \widehat{y}(k) &= H\hat{X}(k) \end{aligned} \tag{8.53}$$

Using this classical form of the state-space model, the steady-state output can be predicted as

$$y(k+\infty|k) = S_\infty\widehat{\mathbf{u}} + s_\infty u(k-1) + y(k) - HI\hat{X}(k) \tag{8.54}$$

$$= S_\infty\widehat{\mathbf{u}} + s_\infty u(k-1) + x(k) \tag{8.55}$$

where s_∞ is the steady-state gain and

$$S_\infty = \begin{bmatrix} s_\infty & s_\infty & \cdots & s_\infty \end{bmatrix}_{1\times M} \tag{8.56}$$

The next subsection describes how to calculate the steady-state gain, s_∞.

8.5.2 Calculation of steady-state Gain

The transfer function corresponding to the state-space model in (8.53) is

$$G(z) = H(\Phi - zI)^{-1}\theta \tag{8.57}$$

where $(zI - \Phi)$ must be invertible i.e. the process is linear and stable. Since the model (8.53) contains an integrator, Φ must be the integrator-free system matrix and θ consists of the impulse response coefficients rather than step response coefficients

Rewrite the integrator-free state-space form of the ARMarkov model in (8.23) as

$$\begin{aligned} \hat{X}_{arm}(k+1) &= \Phi_1 \hat{X}_{arm}(k) + \theta_1 u(k) \\ \hat{y}(k) &= H_{arm} \hat{X}_{arm}(k) \end{aligned} \tag{8.58}$$

and the transfer function corresponding to the state-space model as

$$G(z) = H_{arm} (\Phi_1 - zI)^{-1} \theta_1 \tag{8.59}$$

where Φ_1 must be the integrator-free system matrix and θ_1 consists of the impulse response coefficients. Therefore, $(zI - \Phi_1)$ is invertible. At steady-state $z = 1$ and the steady-state gain of the process can be calculated as

$$s_\infty = G(1) = H_{arm} (\Phi_1 - I)^{-1} \theta_1 \tag{8.60}$$

Using matrix properties

$$(\Phi_1 - I)^{-1} = \frac{1}{|(\Phi_1 - I)|} \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1(\mu+n-1)} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2(\mu+n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{(\mu+n-1)1} & \alpha_{(\mu+n-1)2} & \cdots & \alpha_{(\mu+n-1)(\mu+n-1)} \end{bmatrix}$$

where α_{ij}'s are the elements of the adjoint of matrix $(\Phi_1 - I)$.

For a third order ARMarkov model i.e. $n = 3$,

$$\begin{aligned} \Phi_1 &= \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ -\alpha_2 & -\alpha_1 & 0 & \cdots & 0 & 1 \\ -\alpha_3 & 0 & 0 & \cdots & \cdots & 0 \end{bmatrix}_{(\mu+2)\times(\mu+2)}, \\ \theta_1 &= [h_1, h_2, \cdots, h_{\mu-1}, (\beta_1 - \alpha_1 h_1), \beta_2, \beta_3]^T \\ H_{arm} &= [1, 0, \cdots, 0]_{1\times(\mu+2)} \end{aligned}$$

For this structure of the state-space ARMarkov model, $H_{arm} (\Phi_1 - I)^{-1}$ is the first row of $(\Phi_1 - I)^{-1}$. Therefore, only the first row of the adjoint matrix is calculated. Moreover, it is shown in the following calculations that the determinant and the components of the adjoint matrix have a very simplified structure for the ARMarkov model.

For the third order ARMarkov model,

$$(\Phi_1 - I) = \begin{bmatrix} -1 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ -\alpha_2 & -\alpha_1 & 0 & \cdots & -1 & 1 \\ -\alpha_3 & 0 & 0 & \cdots & \cdots & -1 \end{bmatrix}$$

Following matrix partitioning, the determinant of the matrix becomes

$$|(\Phi_1 - I)| = -(1 + \alpha_1 + \alpha_2 + \alpha_3)$$

Similarly, for an n^{th} order model,

$$|(\Phi_1 - I)| = -(1 + \alpha_1 + \cdots + \alpha_n)$$

The elements of the adjoint matrix also have a simplified form. For example, for the 3^{rd} order model,

$$\begin{aligned} \alpha_{11} &= |a_{11}| \\ \text{where} \quad a_{11} &= \begin{bmatrix} -1 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ -\alpha_1 & 0 & 0 & \cdots & -1 & 1 \\ 0 & 0 & \cdots & \cdots & 0 & -1 \end{bmatrix} \\ \text{and} \quad |a_{11}| &= 1 + \alpha_1 \end{aligned}$$

Similarly, $\alpha_{12} = 1$, and $\alpha_{1i} = 1$ for $i = 2, \cdots, (\mu + n - 1)$. For an n^{th} order model, $|a_{11}| = 1 + \alpha_1$,and $\alpha_{1i} = 1$ for $i = 2, \cdots, (\mu + n - 1)$. Therefore, the steady-state gain is

$$s_\infty = H_{arm}(\Phi_1 - I)^{-1}\theta_1 = \frac{1}{(1 + \alpha_1 + \cdots + \alpha_n)} \begin{bmatrix} 1 + \alpha_1 & 1 & 1 & \cdots & 1 \end{bmatrix} \theta_1 \quad (8.61)$$

Remark 8.1 *The steady-state gain calculated from the special structure state-space model equals the steady-state gain directly calculated from the ARMarkov transfer function model. This is intuitive from the fact that the state space ARMarkov model is formulated in the time-domain. However, it is presented in the following calculations to demonstrate the simplicity and usefulness of the special-structure, state space ARMarkov model.*

Transfer function of the ARMarkov time domain representation in (8.23) is

$$\widehat{G}(z) = \frac{h_1 + h_2 z^{-1} + \cdots + h_{\mu-1} z^{-\mu+1} + \beta_1 z^{-\mu} + \cdots + \beta_n z^{-\mu-n+1}}{\alpha_1 z^{-\mu} + \alpha_2 z^{-\mu-1} + \cdots + \alpha_n z^{-\mu-n+1}}$$

Substituting $z = 1$, the steady-state gain can be written as

$$s_\infty = \widehat{G}(1) = \frac{1}{\alpha_1 + \alpha_2 + \cdots + \alpha_n} [h_1 + h_2 + \cdots + h_{\mu-1} + \beta_1 + \cdots + \beta_n] \qquad (8.62)$$

In (8.61), substituting the elements of θ_1,

$$\begin{aligned} \left[\, 1+\alpha_1 \;\; 1 \;\; 1 \;\; \cdots \;\; 1 \,\right] \theta_1 &= h_1(1+\alpha_1) + h_2 + \cdots + h_{\mu-1} + (\beta_1 - \alpha_1 h_1) + \cdots + \beta_n \\ &= [h_1 + h_2 + \cdots + h_{\mu-1} + \beta_1 + \cdots + \beta_n] \end{aligned}$$

Therefore, from (8.61) and (8.62),

$$s_\infty = G(1) = \widehat{G}(1)$$

8.5.3 Control Calculations Using Steady-state Error Weighting

Assume the predicted steady-state output is augmented into the predicted output vector in (8.10). The augmented state-space model can be written as

$$\begin{aligned} \hat{X}_a(k+1) &= \Phi_a \hat{X}_a(k) + \theta_a \Delta u(k) \\ \hat{y}(k) &= H_a \hat{X}_a(k) \end{aligned} \qquad (8.63)$$

where

$$\Phi_a = \begin{bmatrix} \Phi_{arm} & 0 \\ 0^T & 1 \end{bmatrix}, \; \hat{X}_a(k) = \begin{bmatrix} \hat{X}_{arm}(k) \\ \hat{y}_m^*(k+\infty|k) \end{bmatrix}$$

$$\theta_a = \begin{bmatrix} \theta \\ s_\infty \end{bmatrix}, \; H_a = \begin{bmatrix} H_{arm} & 0 \end{bmatrix}, \; \Phi_{pa} = \begin{bmatrix} \Phi_p & 0 \\ 0^T & 1 \end{bmatrix}_{(N_2+1)\times(\mu+n+1)}$$

and $\hat{y}_m^*(k+\infty|k)$ is the contribution of all past control actions to the predicted steady-state output. The dynamic matrix, G_a can be rebuilt by using the previous matrix G and a new coefficient vector, S_∞ as

$$G_a = \begin{bmatrix} G \\ S_\infty \end{bmatrix}$$

Saudagar (1995) showed that the steady-state gain, s_∞ is not affected by the control horizon, M and hence it does not make sense to use S_∞ in the form of (8.56). A simplified form of the augmented predicted output and the objective function are given by

$$\begin{aligned} Y_\infty(k) &= \begin{bmatrix} Y_{N_2}(k) \\ \hat{y}(k+\infty|k) \end{bmatrix} \\ J &= \left(Y_{sp(a)} - Y_\infty(k)\right)^T Q_\infty \left(Y_{sp(a)} - Y_\infty(k)\right) + \hat{\mathbf{u}}^T \Lambda_\infty \hat{\mathbf{u}} \end{aligned} \tag{8.64}$$

where

$$\begin{aligned} Y_{sp(a)} &= \begin{bmatrix} Y_{sp} \\ y_{sp(\infty)} \end{bmatrix} \\ Q_\infty &= \begin{bmatrix} Q & 0 \\ 0 & \gamma_\infty \end{bmatrix} \end{aligned}$$

The new dynamic matrix is defined as

$$\begin{aligned} G_\infty &= \begin{bmatrix} G \\ x_\infty \end{bmatrix}_{(N_2+1)\times M} \\ \text{where} \quad x_\infty &= \left[s_\infty, 0_{1\times(M-1)}\right]_{1\times M} \end{aligned}$$

Minimizing the objective function, J in (8.64), solution of the control law is

$$\begin{aligned} \hat{\mathbf{u}} &= \left[G_\infty^T Q_\infty G_\infty + \Lambda_\infty\right]^{-1} G_\infty^T Q_\infty E_\infty \\ \text{where} \quad E_\infty &= \left(Y_{sp(a)} - Y_\infty(k)\right) = \begin{bmatrix} E \\ e_\infty \end{bmatrix} \\ e_\infty &= y_{sp(\infty)} - \hat{y}(k+\infty|k) \end{aligned} \tag{8.65}$$

8.5.4 Simultaneous Steady-state and Control Weighting

Variables in the control solution in (8.65) can be expanded as follows:

$$\begin{aligned} G_\infty^T Q_\infty G_\infty &= \begin{bmatrix} G^T & x_\infty^T \end{bmatrix} \begin{bmatrix} Q & 0 \\ 0 & \gamma_\infty \end{bmatrix} \begin{bmatrix} G \\ x_\infty \end{bmatrix} \\ &= G^T Q G + x_\infty^T \gamma_\infty x_\infty \end{aligned}$$

$$G_\infty^T Q_\infty G_\infty + \Lambda_\infty = G^T Q G + x_\infty^T \gamma_\infty x_\infty + \Lambda_\infty$$

The least-squares problem can, then, be written as

$$\begin{aligned} \left[G^T Q G + x_\infty^T \gamma_\infty x_\infty + \Lambda_\infty\right] \hat{\mathbf{u}} &= \begin{bmatrix} G^T & x_\infty^T \end{bmatrix} Q_\infty E_\infty \\ &= \begin{bmatrix} G^T & x_\infty^T \end{bmatrix} \begin{bmatrix} Q & 0 \\ 0 & \gamma_\infty \end{bmatrix} \begin{bmatrix} E \\ e_\infty \end{bmatrix} \end{aligned} \tag{8.66}$$

$$\left[G^T Q G + x_\infty^T \gamma_\infty x_\infty + \Lambda_\infty\right] \hat{\mathbf{u}} = G^T Q E + \gamma_\infty x_\infty^T e_\infty \tag{8.67}$$

Here

$$x_\infty^T \gamma_\infty x_\infty = \gamma_\infty \begin{bmatrix} s_\infty^2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{8.68}$$

$$\gamma_\infty x_\infty^T e_\infty = \gamma_\infty \begin{bmatrix} s_\infty \\ 0 \\ \vdots \\ 0 \end{bmatrix} e_\infty \tag{8.69}$$

Solution to the control problem is

$$\hat{\mathbf{u}} = A_1^* \left(G^T Q E + \gamma_\infty x_\infty^T e_\infty \right)$$

where

$$A_1^* = \left[G^T Q G + x_\infty^T \gamma_\infty x_\infty + \Lambda_\infty \right]^{-1} \tag{8.70}$$

Following the receding horizon principle, only the first control move, $\Delta u(k)$ is implemented. Therefore

$$\begin{aligned} \Delta u(k) &= C^T \hat{\mathbf{u}} = C^T A_1^* \left(G^T Q E + \gamma_\infty x_\infty^T e_\infty \right) \\ &= a^* \left(G^T Q E + \gamma_\infty x_\infty^T e_\infty \right) \\ &= a^* G^T Q E + a_{11}^* s_\infty \gamma_\infty e_\infty \end{aligned} \tag{8.71}$$

where $C^T = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}_{1 \times M}$ and hence a^* is the first row of A_1^*.

The least-squares problem in (8.67) is affected by the steady-state weighting in two terms. The first term on the left side is $\gamma_\infty s_\infty^2$ which can be added to $G^T Q G$ and thereby related to the steady-state weighting or it can be combined with Λ_∞ and considered as part of the control weighting since it can be added to the first component of Λ_∞ as expanded in (8.68). The second term in (8.67) on the right side is $\gamma_\infty s_\infty$ (as expanded in (8.69)) affects only the predicted steady-state error. If the latter term is omitted and the former term is added, the effect is entirely due to the control weighting. If the value γ_∞ remains the same in both the terms, the effect is the same as the complete steady-state weighting problem.

Now if γ_∞ is splitted into two terms and $\gamma_{\infty 1}$ is omitted in the right side of equation (8.67), it can be rewritten as

$$\gamma_\infty = \gamma_{\infty 1} + \gamma_{\infty 2} \tag{8.72}$$

$$\left[G^T Q G + \gamma_{\infty 2} x_\infty^T x_\infty + \left(\gamma_{\infty 1} x_\infty^T x_\infty + \Lambda_\infty\right)\right] U = G^T Q E + x_\infty^T \gamma_{\infty 2} e_\infty \tag{8.73}$$

and the first control move $\Delta u(k)$ in (8.71) is

$$\Delta u(k) = a^* G^T Q E + a_{11}^* s_\infty \gamma_{\infty 2} e_\infty \tag{8.74}$$

Keeping γ_∞ constant and adjusting $\gamma_{\infty 2}$ will tune the controller via both steady-state weighting and control weighting since, for the steady-state weighting $\gamma_{\infty 2}$, control weighting can be written as

$$\Lambda_{\infty 1} = \left(\gamma_{\infty 1} x_\infty^T x_\infty + \Lambda_\infty\right) \tag{8.75}$$

$$= \gamma_{\infty 1} \begin{bmatrix} s_\infty^2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} + \Lambda_\infty \tag{8.76}$$

$$= \begin{bmatrix} \left(\gamma_\infty - \gamma_{\infty 2}\right) s_\infty^2 + \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1M} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{M1} & \lambda_{M2} & \cdots & \lambda_{MM} \end{bmatrix} \tag{8.77}$$

Therefore, the first element is $\lambda_{\infty 11} = \left(\gamma_\infty - \gamma_{\infty 2}\right) s_\infty^2 + \lambda_{11}$ and the other elements in $\Lambda_{\infty 1}$ remain the same as in Λ_∞. Hence, increasing $\gamma_{\infty 2}$ from 0 to 1 (i.e. no weighting and full weighting respectively, as normalized in (Saudagar 1995)) will decrease the control weighting (on the first and implementable control element) by $\gamma_{\infty 2} s_\infty^2$.

Remark 8.2 *If γ_∞ is kept constant, the inverted matrix A_1^* remains the same irrespective of any change in $\gamma_{\infty 2}$. There is no need for matrix re-inversion for the simultaneous steady-state and control tuning i.e. on-line tuning is obtained by adding a single term, $a_{11}^* s_\infty \gamma_{\infty 2} e_\infty$ in the control calculation in (8.74) and hence does not require large additional computational effort. This flexibility in controller tuning using the steady-state error weighting is discussed in more detail in Chapter 11.*

8.5.5 Feedback Gain Update Using $\gamma_{\infty 2}$

Equation (8.74) is rewritten as

$$\begin{aligned}
\Delta u(k) &= C^T A_1^* \left[\begin{array}{cc} G^T Q & x_\infty^T \gamma_{\infty 2} \end{array} \right] \left[\begin{array}{c} E \\ e_\infty \end{array} \right] \\
&= C^T A_1^* \left[\begin{array}{cc} G^T Q & x_\infty^T \gamma_{\infty 2} \end{array} \right] \left[Y_{sp(a)} - \Phi_{N_2 a} X_a \right] \\
&= C^T A_1^* \left[\begin{array}{cc} G^T Q & x_\infty^T \gamma_{\infty 2} \end{array} \right] Y_{sp(a)} - C^T A_1^* \left[\begin{array}{cc} G^T Q & x_\infty^T \gamma_{\infty 2} \end{array} \right] \Phi_{N_2 a} X_a
\end{aligned}$$

The ARM-MPC feedback gain can be written as

$$\begin{aligned}
K_{m\infty} &= C^T A_1^* \left[\begin{array}{cc} G^T Q & x_\infty^T \gamma_{\infty 2} \end{array} \right] \Phi_{N_2 a} \\
&= a^* \left[\begin{array}{cc} G^T Q \Phi_p & x_\infty^T \gamma_{\infty 2} \end{array} \right] \\
&= \left[\begin{array}{cc} a^* G^T Q \Phi_p & a^* x_\infty^T \gamma_{\infty 2} \end{array} \right] \\
&= \left[\begin{array}{cc} K_{m\infty 1} & K_{m\infty 2} \end{array} \right]
\end{aligned}$$

where

$$K_{m\infty 2} = A_1^*(1,1)\, s_\infty \gamma_{\infty 2}$$

As described in the previous subsection, for simultaneous steady-state and control weighting, keeping γ_∞ constant while varying $\gamma_{\infty 2}$ means $K_{m\infty 1}$ remains constant and the value of $K_{m\infty 2}$ varies with the variation in $\gamma_{\infty 2}$.

Lemma 8.2 *For a single input system,*

$$\|K_{m\infty}\|_2 \leq \|K_{m\infty 1}\|_2 + \|K_{m\infty 2}\|_2$$

Proof.

$$\begin{aligned}
K_{m\infty}^T K_{m\infty} = &\left[\begin{array}{cc} K_{m\infty 1} & K_{m\infty 2} \end{array} \right]^T \left[\begin{array}{cc} K_{m\infty 1} & K_{m\infty 2} \end{array} \right] \\
&\left[\begin{array}{c} K_{m\infty 1}^T \\ K_{m\infty 2}^T \end{array} \right] \left[\begin{array}{cc} K_{m\infty 1} & K_{m\infty 2} \end{array} \right] \\
= &\left[\begin{array}{cc} K_{m\infty 1}^T K_{m\infty 1} & K_{m\infty 1}^T K_{m\infty 2} \\ K_{m\infty 2}^T K_{m\infty 1} & K_{m\infty 2}^T K_{m\infty 2} \end{array} \right]
\end{aligned}$$

is a symmetric matrix.

Now

$$\|K_{m\infty}\|_2 = \max \sqrt{eig\,(K_{m\infty}^T K_{m\infty})}$$

For a single input system, $K_{m\infty}$ is a vector and it has only one non-zero singular value (Johnson and Wichern 1988).

Therefore,

$$\max \sqrt{eig\,(K_{m\infty}^T K_{m\infty})} = \sqrt{tr\,(K_{m\infty}^T K_{m\infty})}$$
$$= \sqrt{tr\,(K_{m\infty 1}^T K_{m\infty 1}) + tr\,(K_{m\infty 2}^T K_{m\infty 2})}$$
$$= \sqrt{\|K_{m\infty 1}\|_2^2 + \|K_{m\infty 2}\|_2^2}$$

(since both $K_{m\infty 1}$ and $K_{m\infty 2}$ are vectors and each has one non-zero singular value)

$$\textit{Therefore } \|K_{m\infty}\|_2^2 = \left(\max \sqrt{eig\,(K_{m\infty}^T K_{m\infty})}\right)^2$$
$$= \|K_{m\infty 1}\|_2^2 + \|K_{m\infty 2}\|_2^2$$

Since $K_{m\infty} \geq 0$, $K_{m\infty} \geq 0$ and $K_{m\infty} \geq 0$, the following inequality between the feedback gain norms exists:

$$\|K_{m\infty}\|_2 \leq \|K_{m\infty 1}\|_2 + \|K_{m\infty 2}\|_2 \;\; \textit{where} \;\; \|K_{m\infty 2}\|_2 = A_1^*(1,1)\, s_\infty \gamma_{\infty 2}$$

■

Since $\|K_{m\infty 1}\|_2$ remains constant for a constant γ_∞, the feedback gain $\|K_{m\infty 2}\|_2$ can be varied by varying $\gamma_{\infty 2}$ and hence $\|K_{m\infty}\|_2$. For example, for $\gamma_{\infty 2} = 0$, $\|K_{m\infty}\|_2 = \|K_{m\infty 1}\|_2$ and the tuning effect can be interpreted as due to changes in control weighting. For $\gamma_{\infty 2} = \gamma_\infty$, the effect of changing $\gamma_{\infty 2}$ can be interpreted in terms of the steady-state weighting. Using a value of $\gamma_{\infty 2}$ between 0 and γ_∞, results in changes in the feedback gain (and hence in performance) that can be interpreted as due to *simultaneous* changes in steady-state weighting and control weighting. This gives the flexibility of using either control weighting or steady state weighting or a combination of the two weighting parameters using a single tuning knob, $\gamma_{\infty 2}$. The main advantage of this approach is that it is not necessary to change the controller design and repeat the entire calculation.

The role of $\gamma_{\infty 2}$ as a tuning parameter and its usefulness in the predictive controller tuning to obtain performance/robustness trade-off will be discussed in more detail in Chapter 11.

8.6 Examples

The theory developed in the previous sections is illustrated through simulation examples in this section. The ARMarkov identification along with the predictive control design is used with the following third order process

$$G(s) = \frac{1}{(s+1)(3s+1)(5s+1)}$$

After discretization using a sampling interval of 1, the discretized process becomes

$$G(z^{-1}) = \frac{0.0077z^{-1} + 0.0212z^{-2} + 0.0036z^{-3}}{1 - 1.9031z^{-1} + 1.1514z^{-2} - 0.2158z^{-3}} \tag{8.78}$$

Example 8.1

A state space ARM-MPC was designed by using the procedure described in Section 8.53 and an ARMarkov model with order $n = 3$ and the number of Markov parameters in the model $\mu = 10$. The controller tuning parameters used are as follows: $N_2 = 10$, $M = 1$, $Q = R = I$ and $\lambda = 0.0$. The process response is shown in Figure 8.1. As expected, the state space response is equivalent to the response obtained for the same process in Chapter 6 using the input-output ARM-MPC with the same tuning parameters.

Example 8.2

Consider the same process in (8.78). An ARM-MPC was designed using the same parameters as in Example 8.1. The control horizon of the controller was increased on-line using the recursive calculations described in Section 8.60. Set-point tracking performance is shown in Figure 8.2. As the control horizon M, is increased the control action and speed of the output response are increased.

Example 8.3

For the same process in (8.78), the ARM-MPC was designed using the steady state error weighting. The controller tuning parameters used are: $N_2 = 10$, $M = 2$, $Q = R = I$ and $\lambda = 0.0$. Step tracking performance for different $\gamma_{\infty 2}$ is shown in Figure 8.3. As the value of $\gamma_{\infty 2}$ increases, the controller becomes more conservative

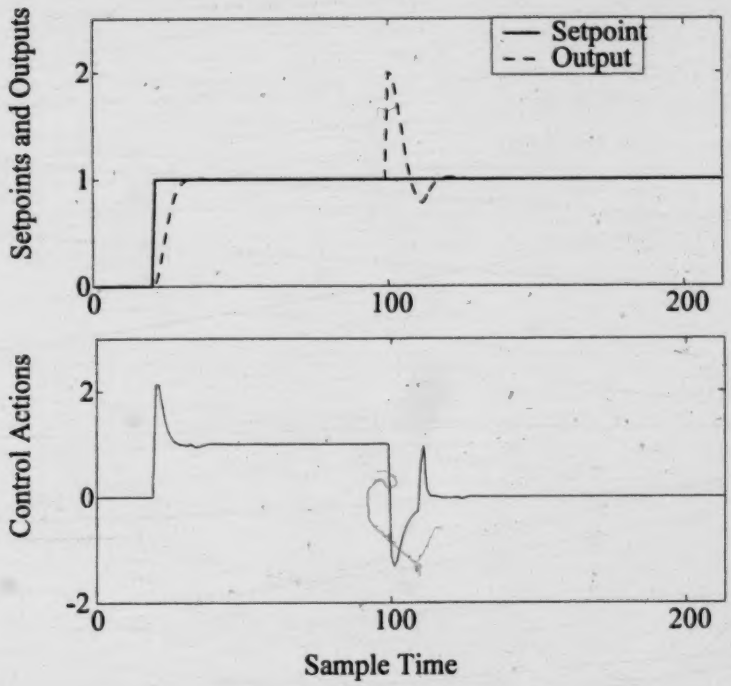


Figure 8.1: Step Tracking and Disturbance Rejection using ARM-MPC with a Type-1 Disturbance Model. A unit step disturbance was added at time 100.

and with $\gamma_{\infty 2} = 1$ i.e. full steady state weighting, the controller gives mean level performance. A combination of steady state weighting and control weighting is obtained via changing a single parameter, $\gamma_{\infty 2}$ and does not require on-line matrix inversion and reduces the computational effort.

8.7 Conclusions

- The state-space formulation of the ARM-MPC developed in this chapter is equivalent to the input-output formulation developed in Chapter 6 but is better suited to classical performance/robustness and stability analysis.

- The states of the special structure state-space model are the predicted present /future outputs which have a physical meanings for the operators of industrial applications. It needs smaller number of Markov parameters than the dual-

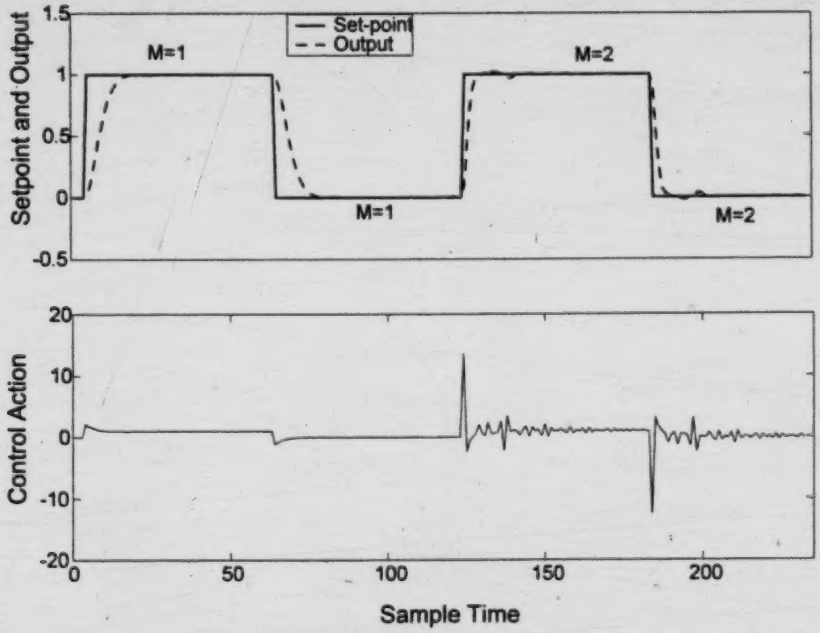


Figure 8.2: Process response by changing control horizon on-line.

model of Qi (1997)due to the presence of the parametric part of the model.

- Recursive control move calculation avoids having to redesign the controller when the control horizon, M is changed as a means of on-line tuning.

- Stability of the closed-loop system is increased by increasing steady-state error weighting used in this chapter. As a limiting case, for pure steady-state error weighting, closed-loop stability is guaranteed for an open-loop stable process.

- The steady state error weighting is interpreted as a combination of steady state weighting and control weighting. It is shown later in Chapter 11 that the controller can be tuned for different performance/robustness properties using a combination of the two weightings without redesigning the controller.

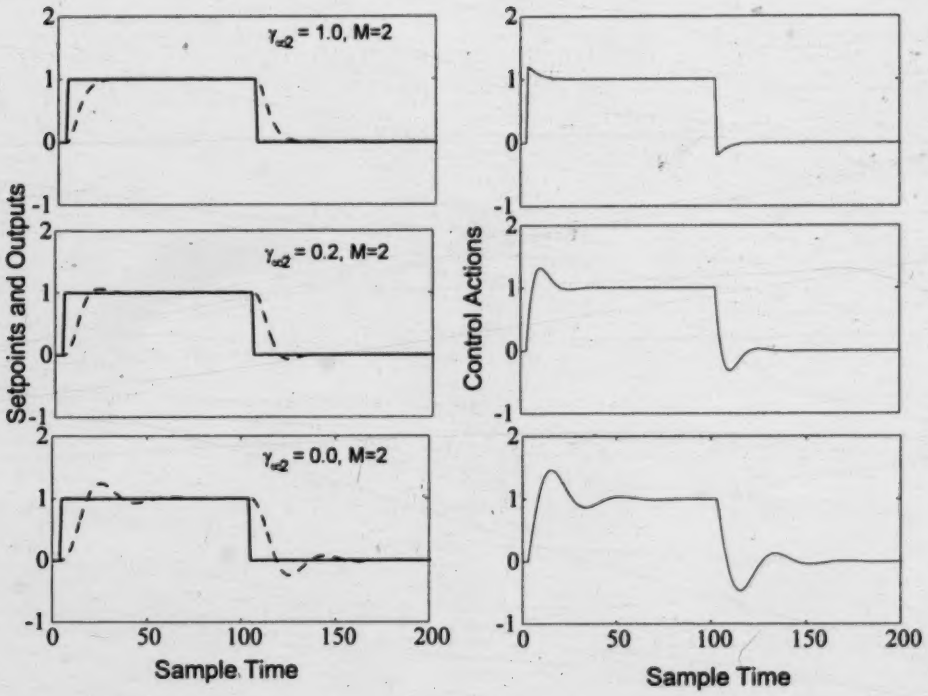


Figure 8.3: Set-point tracking for simultaneous steady state error and control weighting using a single parameter, $\gamma_{\infty 2}$.

Chapter 9

MPC Design Using a Classical State-Space Model Generated from Markov Parameters

9.1 Introduction

Classical state space models have been identified and used in Model-based Predictive Control (MPC) algorithms by many researchers (e.g. (Ricker 1991, Lee *et al.* 1992, Gambier and Unbehauen 1993, Ruscio 1997*b*, Ruscio 1997*a*)). There are numerous methods for the identification of subspace state space models. The basic question in all applications is whether the states of the system can be excited which is related to system controllability and observability. The development of the model or realization of the system involves the computation of the triplet matrices A, B, C. There are an infinite number of realizations that can produce the same response when the system is subjected to a given input. Typically minimal realization of the state space model, originally developed by Ho and Kalman (1965), is used to avoid higher order structures. Minimal realization of the system matrix starts with the construction of the Hankel matrix that consists of the process Markov parameters. It was shown in Chapter 2 of this thesis that the Markov parameters can be directly estimated by the ARMarkov method using process input-output data. Therefore, it is logical to use those Markov parameters to generate a classical state space model and design an MPC using that model. Moreover, it was also shown in Chapter 4 of this thesis that the Markov parameters estimated using the ARMarkov method are, consistent and hence, in the limit as the data vector increases, the Hankel matrix should converge

to its true value and lead to a better estimate of the state space model and in turn, to better MPC performance than other (non-consistent) identification methods.

The Eigensystem Realization Algorithm (ERA) has been used by researchers (e.g. (Juang 1994, Juang and Papa 1985, Juang and Papa 1986)) to develop a state space realization from the Hankel matrix and it was shown that the ERA is, in practice, resilient to the effects of noise corrupting the Markov parameters while developing the minimal realization of the system.

In this chapter, the Markov parameters obtained using the ARMarkov identification method are used to develop the block Hankel matrix and ERA is used to obtain the minimal realization of a classical state space model.

The primary reason for developing an equivalent state space model from the identified Markov parameters is to provide a basis for utilizing the wide variety of state space design, analysis and simulation techniques that are available in the literature. Actual implementation of the resulting MPC can be done using state space methodology or an input-output model-based approach as discussed in earlier chapters. The classical state space model (minimal realization) is developed in Section 9.2. MPC design using the classical state space realization is discussed in Section 9.3. Simulation examples and experimental evaluation of the state space MPC are given in Sections 9.4 and 9.5 respectively.

9.2 State-Space Model From the Markov Parameters

The classical state space expression of a discrete-time, finite-dimensional, linear, time-invariant SISO system is written as

$$x(k+1) = A_x x(k) + B_x u(k), \tag{9.1}$$

$$y(k) = C_x x(k) + D_x u(k) \tag{9.2}$$

where $A_x \in R^{n \times n}, B_x \in R^{n \times 1}, C_x \in R^{1 \times n}$, and $D_x \in R$.

The Markov parameters h_j of this system are defined as

$$\begin{aligned} h_j &\triangleq D_x && \text{for } j = 0 \\ &\triangleq C_x A_x^{j-1} B_x && \text{for } j \geq 1 \end{aligned} \tag{9.3}$$

The time-domain ARMarkov representation of this system, as discussed in Chapter 2, is written as

$$y(k) = -\sum_{j=1}^{n} \alpha_j y(k-\mu-j+1) + \sum_{j=1}^{\mu} h_{j-1} u(k-j+1) + \sum_{j=1}^{n} \beta_j u(k-\mu-j+1) \tag{9.4}$$

where h_j, $j = 0, 1 \cdots, \mu - 1$ are the first μ Markov parameters of the system. The Markov parameters are estimated using the ARMarkov least-squares method discussed in Chapter 2.

9.2.1 Eigensystem Realization Algorithm (ERA)

The first step in using the ERA to develop a state space realization is to build the Markov block Hankel matrix using Markov parameters.

The $(r+1)l \times (s+1)m$ block Hankel matrix with $j \geq 0$ is constructed by stacking the Markov parameters obtained as follows.

$$H_{r,s,j} \triangleq \begin{bmatrix} h_j & \cdots & h_{j+s} \\ \vdots & \ddots & \vdots \\ h_{j+r} & \cdots & h_{j+r+s} \end{bmatrix}_{(r+1)l \times (s+1)m} \tag{9.5}$$

where l, m are the number of outputs and inputs respectively and r, s are arbitrary integers.

For a system with Mcmillan degree n (where Mcmillan degree is defined as the degree of the least common denominator of all minors of a transfer function that is equal to the minimal realization of the transfer function), if $r, s \geq n-1$ and the system is noise free, the rank of $H_{r,s,1}$ is equal to the system order or the dimension of the state matrix (Kailath 1980, Juang 1994). This is a well-known fact resulting from the *controllability* and *observability* matrices (Kailath 1980, Juang 1994) because the Hankel matrix, $H_{r,s,j}$ can be decomposed as

$$H_{r,s,j} = P_o A^j Q_c \tag{9.6}$$

where

$$P_o = \begin{bmatrix} C_x \\ C_xA_x \\ C_xA_x^2 \\ \vdots \\ C_xA_x^{n-1} \end{bmatrix} \text{ and}$$

$$Q_c = \begin{bmatrix} B_x & A_xB_x & A_x^2B_x & \cdots & A_x^nBx \end{bmatrix}$$

P_o and Q_c are the *observability* and *controllability* matrices each with rank equal to the system order, n.

The ERA uses the Singular Value Decomposition (SVD) of the block Hankel matrix, $H_{r,s,1}$ as

$$H_{r,s,1} = R\Sigma S^T \tag{9.7}$$

where the columns of R and S are orthonormal and

$$\Sigma = \begin{bmatrix} \Sigma_n & 0 \\ 0 & 0 \end{bmatrix}$$

where $\Sigma_n = diag\{\sigma_1, \sigma_2, \cdots, \sigma_n\}$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ are the non-zero singular values of the Hankel matrix. Constructing R_n and S_n with the first n columns of R and S respectively, the Hankel matrix can be written as

$$H_{r,s,1} = R_n\Sigma_n S_n^T \tag{9.8}$$

where $R_n^TR_n = S_n^TS_n = I_n$.

The solution for the state matrix, A_x can be written as (Juang 1994)

$$A_x = \Sigma_n^{-1/2}R_n^TH_{r,s,2}S_n\Sigma_n^{-1/2}$$

The triplet realization of the state space system is

$$\hat{A}_x = \Sigma_n^{-1/2}R_n^TH_{r,s,2}S_n\Sigma_n^{-1/2},\ \hat{B}_x = \Sigma_n^{1/2}S_n^TE_{s,m},\ \hat{C}_x = E_{r,l}^TR_n\Sigma_n^{1/2} \tag{9.9}$$

where $E_{s,m} = \begin{bmatrix} I_j \\ 0_{ij\times j} \end{bmatrix} \in \mathrm{R}^{(i+1)j\times j}$, I_j is the identity matrix of order j and $0_{ij\times j}$ is the null matrix of order $ij \times j$. For proofs and more details of the ERA algorithm, see Juang and Papa (1985)and Juang (1994). The minimum number of Markov parameters required to obtain a minimal system realization is $2n+1$ for $r, s = n-1$.

Therefore, in the ARMarkov model, μ must be chosen such that $\mu \geq 2n+1$ (n =order of the ARMarkov parametric model).

In practice, the process data is corrupted by noise and other interference and the rank of $H_{r,s,j}$ is greater than the system order and may be of full rank. Then $diag(\Sigma)$ will contain more than n non-zero singular values. Determining the system order in such cases is discussed next.

9.2.2 Selection of Model Order When Using Noise Corrupted Data

When the data is noise corrupted, the number of non-zero singular values of the block Hankel matrix is greater than the system order or the dimension of the state matrix. In such a case, the system order is determined by examining the magnitude of the singular values and truncating the smaller singular values of $H_{r,s,1}$ while retaining the singular values corresponding to the dominant dynamics of the system. However, the question is which singular values are considered to be insignificant and truncated. Juang and Papa (1986)and Juang (1994)discussed procedures for order determination using noise-corrupted data.

In this chapter, a simple measure is proposed to determine the system order for noise-corrupted data. Assuming $(r+1)l \geq (s+1)m$ in (9.5), the Hankel matrix, $H_{r,s,j}$ has a maximum of $(s+1)m$ non-zero singular values $\sigma_1, \sigma_2, \cdots, \sigma_{(s+1)m}$. Now define

$$\eta = \frac{\sigma_1 + \sigma_2 + \cdots + \sigma_n}{\sum_{i=1}^{(s+1)m} \sigma_i} \times 100 \tag{9.10}$$

as the percentage of the total singular values explained by the first n singular values. η can be specified depending on the user-chosen approximation or truncation error. Suppose $\eta = 95\%$ i.e. n is selected such that the first n singular values explain the 95% of the total singular values. The remaining singular values are considered insignificant and due to the effect of noise corruption and are truncated. The system realization then proceeds as discussed above.

Once the minimal realization of the system $(\hat{A}_x, \hat{B}_x, \hat{C}_x, \hat{D}_x)$ is estimated, an MPC can be designed using the classical state space model as discussed in the next section.

9.3 State-Space MPC Design

Ricker (1991)presented a tutorial style formulation of a model predictive controller using classical state space models. In this chapter, more or less the same formulation is applied to develop state space MPC using the system realization obtained using the Markov parameters identified using the ARMarkov method. More details on the MPC formulation can be found in Ricker (1991).

9.3.1 Augmented State-Space Model

The linear time-invariant state-space model of a process can be expressed as:

$$\begin{aligned} x(k+1) &= A_x x(k) + B_x u(k) + \Gamma_v v(k) + \Gamma_w w(k) \qquad (9.11) \\ y(k) &= \overline{y}(k) + z(k) \\ &= C_x x(k) + z(k) \end{aligned}$$

where $x(k)$ is the state vector, u is the manipulated variable vector, v represents measured disturbances, w is the unmeasured disturbance, z is the measurement noise, and $A_x, B_x, \Gamma_v, \Gamma_w$ and C_x are constant matrices. This model can be expressed in difference variables form as:

$$\begin{aligned} \Delta x(k) &= A_x \Delta x(k-1) + B_x \Delta u(k-1) + \Gamma_v \Delta v(k-1) + \Gamma_w \Delta w(k-1) \qquad (9.12) \\ y(k) &= C_x A_x \Delta x(k-1) + \overline{y}(k-1) + B_x \Delta u(k-1) + C_x \Gamma_v \Delta v(k-1) \\ &\quad + C_x \Gamma_w \Delta w(k-1) + z(k) \qquad (9.13) \end{aligned}$$

where $\Delta x(k) = x(k) - x(k-1)$. Now the states can be augmented with the plant output (before the addition of the measurement noise) as:

$$\begin{aligned} x_a(k+1) &= A_{xa} x_a(k) + B_{xa} \Delta u(k) + \Gamma_{va} \Delta v(k) \\ &\quad + \Gamma_{wa} \Delta w(k) \qquad (9.14) \\ y(k) &= C_{xa} x_a(k) + z(k) \qquad (9.15) \end{aligned}$$

where $x_a = [\Delta x(k)^T \ \overline{y}(k)^T]^T$ and

$$A_{xa} = \begin{bmatrix} A_x & 0 \\ C_x A_x & I \end{bmatrix}, B_{xa} = \begin{bmatrix} B_x \\ C_x B_x \end{bmatrix}, \Gamma_{va} = \begin{bmatrix} \Gamma_v \\ C_x \Gamma_v \end{bmatrix}, \Gamma_{wa} = \begin{bmatrix} \Gamma_w \\ C_x \Gamma_w \end{bmatrix}$$

$$C_{xa} = \begin{bmatrix} 0 & I \end{bmatrix}$$

The difference form of the states is used to avoid the initialization of the process i.e. $\Delta x(0) = 0$ if the process is initially at steady-state.

9.3.2 State Estimation

The generalized approach for augmented state estimation of a feedback system used by Ricker (1991)is:

$$\begin{aligned} \hat{x}_a(k+1/k) &= A_{xa}\hat{x}_a(k/k-1) + B_{xa}\Delta u(k) + \Gamma_{va}\Delta v(k) \\ &\quad + K[y(k) - \hat{y}_m(k/k-1)] \\ \hat{y}(k/k-1) &= C_{xa}\hat{x}_a(k/k-1) \end{aligned} \tag{9.16}$$

where $\hat{x}_a(k+1/k)$ is an estimate of the augmented plant state, $\hat{y}(k/k-1)$ is the prediction of the noise-free plant output, $y(k)$ is the measured plant output, $\hat{y}_m(k/k-1)$ is the first n_{ym} elements of $\hat{y}(k/k-1)$ and K is the estimator gain.

The estimator gain is

$$K = \begin{bmatrix} K_1 \\ K_2 \end{bmatrix}$$

where K_1, K_2 are of n by n_{ym} and n_y by n_{ym} respectively.

9.3.3 Output Prediction and Control Move Calculation

Using (9.16), for a prediction horizon N_2, predicted future outputs over the specified horizon can be written as:

$$\hat{\mathbf{Y}}(k) = \mathbf{S}_x\Delta\hat{x}(k/k-1) + \mathbf{1}\hat{y}(k/k-1) + G\hat{\mathbf{u}}(k) + \mathbf{S}_v\mathbf{v}(k) + \mathbf{S}_y[y(k) - \hat{y}_m(k/k-1)] \tag{9.17}$$

where the predicted future output vector is

$$\hat{\mathbf{Y}}(k) = \begin{bmatrix} \hat{y}(k+1/k) \\ \hat{y}(k+2/k) \\ \vdots \\ \hat{y}(k+N_2/k) \end{bmatrix},$$

the present/future control vector is,

$$\hat{\mathbf{u}} \triangleq \begin{bmatrix} \Delta u(k/k) \\ \Delta u(k+1/k) \\ \vdots \\ \Delta u(k+M-1/k) \end{bmatrix},$$

the dynamic matrix is,

$$G = \begin{bmatrix} C_x B_x & 0 & \cdots & 0 \\ C_x(A_x+I)B_x & C_x B_x & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ C_x(\Sigma_{i=1}^{N_2} A_x^{i-1})B_x & C_x(\Sigma_{i=1}^{N_2-1} A_x^{i-1})B_x & \cdots & C_x B_x \end{bmatrix},$$

and

$$\mathbf{v}(k) \triangleq \begin{bmatrix} \Delta v(k/k) \\ \Delta v(k+1/k) \\ \vdots \\ \Delta v(k+N_2-1/k) \end{bmatrix}, \mathbf{S}_x = \begin{bmatrix} C_x A_x \\ C_x(A_x^2+A_x) \\ \vdots \\ \sum_{i=1}^{N_2} C_x A_x^i \end{bmatrix}, \mathbf{1} = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix}$$

$$\mathbf{S}_v = \begin{bmatrix} C_x\Gamma_v & 0 & \cdots & 0 \\ C_x(A_x+I)\Gamma_v & C_x\Gamma_v & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ C_x(\Sigma_{i=1}^{N_2} A_x^{i-1})\Gamma_v & C_x(\Sigma_{i=1}^{N_2-1} A_x^{i-1})\Gamma_v & \cdots & C_x\Gamma_v \end{bmatrix},$$

$$\mathbf{S}_y = \begin{bmatrix} K_2 \\ C_x A_x K_1 + K_2 \\ \vdots \\ \sum_{i=1}^{N_2} C_x A_x^{i-1} K_1 + K_2 \end{bmatrix}$$

Assume that the future values of the measured disturbance, v will be equal to the present value (as done in DMC). Then $\Delta v(k+1/k) = \Delta v(k+2/k) = \cdots \Delta v(k+N_2-1/k) = 0$ and equation (9.17) becomes

$$\hat{\mathbf{Y}}(k) = \mathbf{S}_x \Delta\hat{x}(k/k-1) + \mathbf{1}\hat{y}(k/k-1) + G\hat{\mathbf{u}}(k) + \mathbf{S}_{v0}\Delta v(k) + \mathbf{S}_y[y(k) - \hat{y}_m(k/k-1)] \tag{9.18}$$

where

$$\mathbf{S}_{v0} \triangleq \left[[C_x\Gamma_v]^T \quad [C_x(A_x+I)\Gamma_v]^T \quad \cdots \quad [C_x(A_x^{N_2-1} + A_x^{N_2-2} + \cdots + I)\Gamma_v]^T\right]^T$$

The control objective is to determine the manipulated variable, $\hat{\mathbf{u}}(k)$ by minimizing the quadratic objective function,

$$\min_{\hat{\mathbf{u}}} J(k) = \left[\mathbf{r}(k) - \hat{\mathbf{Y}}(k)\right]^T Q \left[\mathbf{r}(k) - \hat{\mathbf{Y}}(k)\right] + \hat{\mathbf{u}}(k)^T R\hat{\mathbf{u}}(k) \tag{9.19}$$

where Q and R are output and input weighting matrices respectively and

$$\mathbf{r}(k) = \left[r(k+1/k)^T \; r(k+2/k)^T \; \cdots \; r(k+N_2/k)^T\right]^T \tag{9.20}$$

is the future output set-point vector.

The solution of the above unconstrained optimization problem is the control law,

$$\hat{\mathbf{u}}(k) = (G^TQG + R)^{-1}G^TQ\left[\mathbf{r}(k) - \mathbf{f}(k)\right] \tag{9.21}$$

$$= K_{mpc}\left[\mathbf{r}(k) - \mathbf{f}(k)\right] \tag{9.22}$$

where the future free response vector,

$$\mathbf{f}(k) = \mathbf{S}_x\Delta\hat{x}(k/k-1) + \mathbf{1}\hat{y}(k/k-1) + \mathbf{S}_{v0}\Delta v(k) + \mathbf{S}_y[y(k) - \hat{y}_m(k/k-1)] \tag{9.23}$$

is obtained by setting $\hat{\mathbf{u}}(k) = \mathbf{0}$ in equation (9.18) i.e. keeping the future control moves constant over the prediction horizon. The controller gain matrix

$$K_{mpc} = (G^TQG + R)^{-1}G^TQ$$

is constant for a given set of controller tuning parameters N_2, M, Q, R and can be calculated off-line.

9.3.4 Feedback Observer or State Estimator Design

The estimator gain matrix, K can be calculated using the conventional approach based on a state space model and the solution of a Riccati equation. If the noise

terms $\Delta w(k)$ and $z(k)$ in (9.14) and (9.15) are white noise with known mean and covariances, then the estimator gain, K is the steady state Kalman filter gain and can be calculated as

$$K = A_{xa} P_x C_{xa}^T \left(R_2 + C_{xa} P_x C_{xa}^T\right)^{-1} \tag{9.24}$$

where P_x is the steady state solution of the Riccati equation

$$P_x(k+1) = A_{xa} P_x(k) A_{xa}^T + R_1 - A_{xa} P_x(k) C_{xa}^T \left(R_2 + C_{xa} P_x C_{xa}^T\right)^{-1} C_{xa} P_x(k) A_{xa}^T \tag{9.25}$$

R_1, R_2 are the covariance matrices of $\Delta w(k)$ and $z(k)$ respectively. Ricker (1991)discussed estimator design using different types of disturbance models e.g. type 1 disturbances, type 2 disturbances and so on. In the simplest form, for the DMC type of disturbances, $K_1 = 0$ and $K_2 = I$. Therefore, $K = [0; I]$ and $\mathbf{S}_y = \mathbf{1}$.

9.3.5 Closed-loop Formulation of the State-space MPC

Using the receding horizon principle, the calculated current control move is

$$\Delta u(k) = C^T \hat{\mathbf{u}}(k) \tag{9.26}$$

A closed-loop state-space formulation is shown in block diagram form in Figure 9.1 (Ricker 1991).

The matrices in the block diagram are derived using equations (9.14)-(9.26) and can be written as

$$A_1 = \begin{bmatrix} A_x & \mathbf{0} \\ C_x A_x & I \end{bmatrix}, B_1 = \begin{bmatrix} B_x & \Gamma_v \\ C_x B_x & C_x \Gamma_v \end{bmatrix}, C_1 = \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix}, D_1 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{9.27}$$

$$A_2 = \begin{bmatrix} A_x & -K_1 \\ C_x A_x & I - K_2 \end{bmatrix}, B_2 = \begin{bmatrix} K_1 \\ K_2 \end{bmatrix}, C_2 = \begin{bmatrix} \mathbf{S}_x & \mathbf{1} - \mathbf{S}_y \end{bmatrix}, D_{21} = \mathbf{S}_y \tag{9.28}$$

$$A_3 = \begin{bmatrix} A_x - B_x K_{mpc} \mathbf{S}_x & -B_x K_{mpc} \mathbf{1} \\ C_x (A_x - B_x K_{mpc} \mathbf{S}_x) & \mathbf{I} - B_x K_{mpc} \mathbf{1} \end{bmatrix}, \tag{9.29}$$

$$B_3 = \begin{bmatrix} B_x K_{mpc} & \Gamma_v - B_x K_{mpc} \mathbf{S}_{v0} \\ C_x B_x K_{mpc} & C_x (\Gamma_v - B_x K_{mpc} \mathbf{S}_{v0}) \end{bmatrix}; \tag{9.30}$$

$$C_3 = \begin{bmatrix} -K_{mpc} \mathbf{S}_x & -K_{mpc} \mathbf{1} \end{bmatrix}, D_3 = \begin{bmatrix} K_{mpc} & -K_{mpc} \mathbf{S}_{v0} \end{bmatrix} \tag{9.31}$$

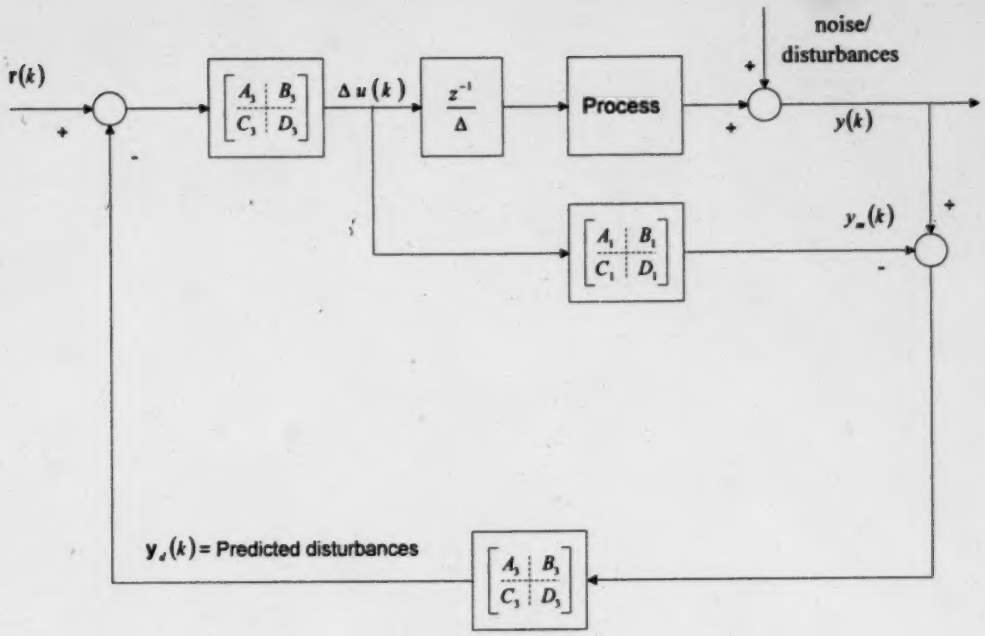


Figure 9.1: Closed-loop MPC system with disturbance predictor.

9.4 Simulation Results

Consider the third order process discussed in Chapter 2.

$$G(z) = \frac{0.0077z^{-1} + 0.0212z^{-2} + 0.0036z^{-3}}{1 - 1.9031z^{-1} + 1.1514z^{-2} - 0.2158z^{-3}} \tag{9.32}$$

The proposed ARMarkov/ERA algorithm was applied to this process to develop a state space model. The step response of the estimated state space model is very close to the actual process step response (Figure 9.2). Performance of the state space MPC is shown in Figure 9.3. As expected, the step tracking and disturbance rejection (at sampling step $t = 50$ and $t = 150$) performance of the state space MPC is very good.

9.5 Experimental Evaluation of Classical MPC on a Pilot Scale Process

An MPC using a general state space model developed from the Markov parameters obtained using the ARMarkov method was implemented on a pilot scale Continuous

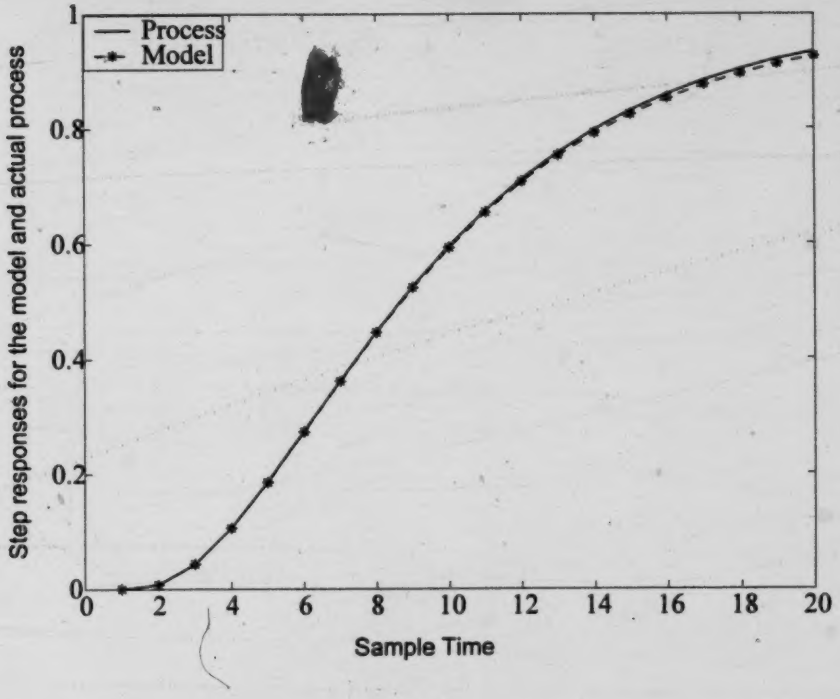


Figure 9.2: Step resposes for the actual process and the state space model.

Stirred Tank Heater (CSTH) in the Computer Process Control (CPC) laboratory in the Chemical & Materials Engineering department at the University of Alberta. The CSTH is described in Chapter 2 and a schematic diagram of the process is shown in Figure 2.3. Water level in the tank and the water temperature were selected as the two controlled variables and the valve openings (manipulating cold water flow and steam flow) were selected as the two manipulated variables. The process considered is a 2×2 MIMO system. However, the water level in the tank varies only with the inlet water flow and is invariant to the steam flow rate.

Markov parameters were determined for the above process using open-loop input-output data and the ARMarkov identification method described in Chapter 2. These Markov parameters were then used to develop a classical state space model following the procedure discussed in Section 9.2 and an MPC was designed as discussed in Section 9.3. The ARMarkov model order was $n = 2$ and the number of Markov parameters in the model was assumed to be $\mu = 26$. The controller parameters

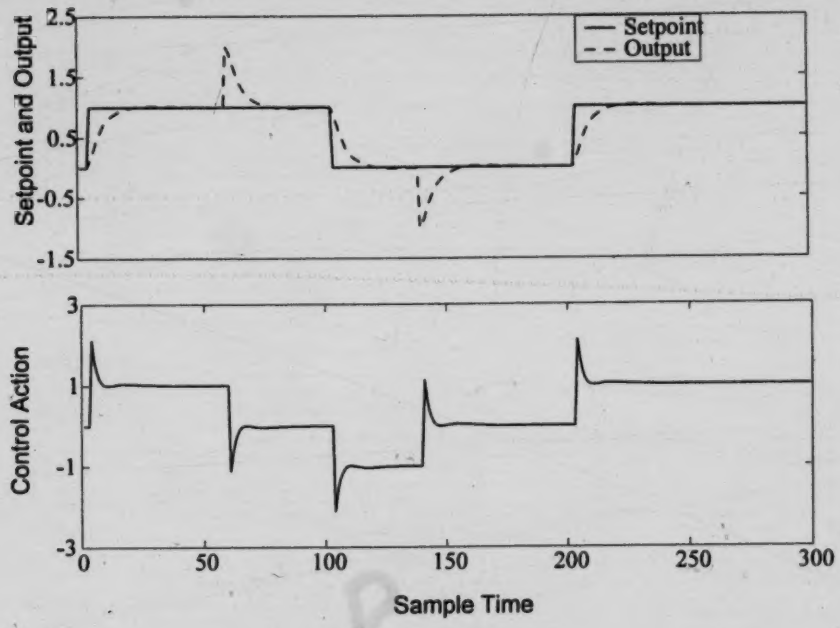


Figure 9.3: Performance of the state space MPC developed using the ARMarkov/ERA model.

used were as follows: $N_2 = 10$, $M = 1$, $Q = R = I$ and $\lambda = 0.0$. The inputs and outputs were sampled every four seconds. The CSTH responses shown in Figure 9.4 are comparable to those obtained using other advanced control techniques and are considered acceptable.

9.6 Conclusions

- Consistent Markov parameters identified using the ARMarkov procedure discussed in Chapter 2 were used to build a *classical* state space model. An Eigensystem Realization Algorithm (ERA) was used for building the A, B, C, D matrices because it is based on decomposition of the Hankel matrix constructed from the estimated Markov parameters.

- Users who wish to use *classical* state space theory/algorithms for part of the design, analysis or implementation of their MPC, can do so without having to do a separate identification for the $[A, B, C, D]$ matrices and will have a guarantee

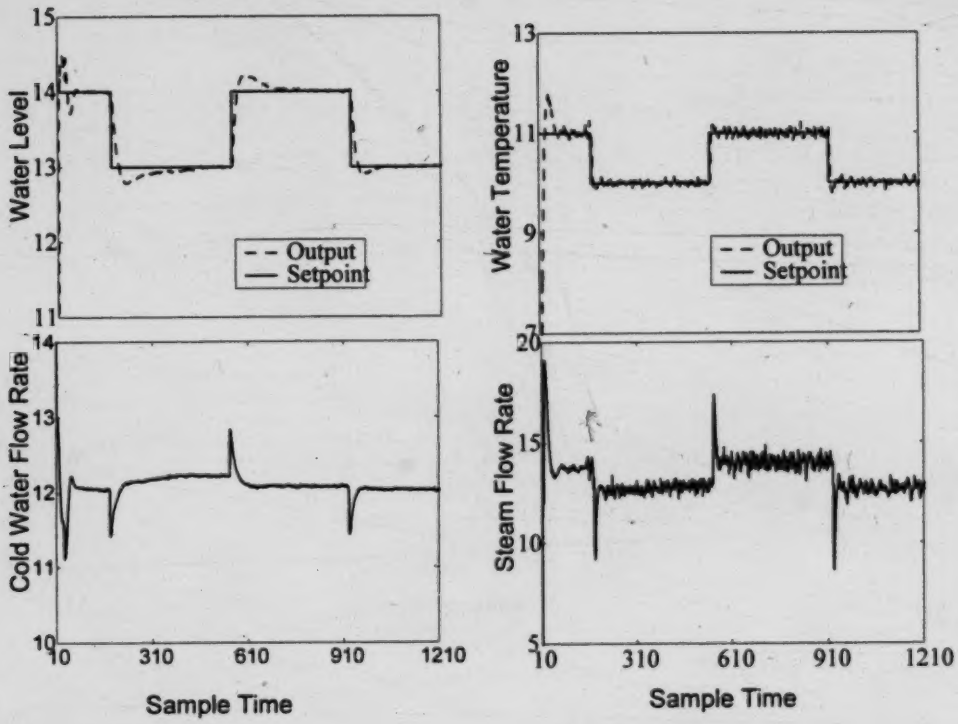


Figure 9.4: Step tracking performance of state space MPC implemented on a pilot scale CSTH (2×2 MIMO system).

of the equivalence of the input-output and state space models.

- The set point tracking and disturbance rejection performance of the state space MPC were illustrated by numerical simulations and experimental results from a computer controlled pilot scale process.

Part III

Analysis of ARM-MPC

Analysis of the performance/robustness properties of the closed-loop system as well as on-line tuning of the designed ARM-MPC are discussed in this part of the thesis. Chapter 10 shows that ARMarkov identification using closed-loop (CL) input-output data provides multiple metrics for the first level of performance assessment i.e. a minimum variance performance benchmark, time-domain performance measures in terms of CL settling time, rise time, overshoot etc., and frequency domain metrics such as sensitivity functions of the closed loop system. An LQG performance benchmark is recommended for use as a user-specified benchmark for the performance assessment of ARM-MPC and is discussed in Chapter 10. Robustness analysis using matrix perturbation techniques, also done in Chapter 10, shows that the *special structure dual-model state space ARM-MPC* leads to robustness bounds that are less conservative than those obtained using the Small Gain Theorem (SGT). On-line tuning of ARM-MPC using a single parameter α is discussed in Chapter 11. The $0 \leq \alpha \leq 1$ parameter combines the outputs of two different MPC controllers to obtain user specified performance/robustness trade-offs. The two individual controllers can be designed such that the second controller output is obtained by updating the output of the first controller and thereby avoid additional computation needed for matrix inversion and/or Diophantine equation solution at every sampling interval. A surface of different controller outputs is constructed using the outputs from three different controllers with specific (user defined) performance/robustness properties. This tuning surface allows a wide range of tuning scope without much additional computation.

Chapter 10

Performance and Robustness Analysis of ARM-MPC

10.1 Introduction

The ultimate goal of designing a controller is to produce good performance over the expected range of operating conditions and uncertainties. When a controller is implemented on a process, it is important to determine how well the controller is working and if it is giving the performance it is designed for. There are many methods of measuring the performance of a designed system but the most practical are those that use closed-loop process data. The desired performance is specified by a performance criterion or a benchmark and the actual performance is then measured and assessed against this criterion.

10.1.1 Performance Assessment

To assess or monitor control loop performance, there must be some measure or benchmark against which the actual controller performance can be assessed. DeVries and Wu (1978) used one step ahead prediction error as the performance benchmark. Åstrom (1970), Harris (1989), Desborough and Harris (1993) and Stanfelj *et al.* (1993) used Minimum Variance (MV) control performance as the benchmark. Huang *et al.* (1997*a*) and Harris *et al.* (1996) extended the MV performance benchmark to MIMO systems. Huang *et al.* (2000) developed performance assessment techniques for MIMO feedback plus feedforward controllers and applied them to industrial cases. Kozub (1996) proposed user specified performance benchmarks in terms of closed-loop set-

tling time, overshoot etc. which are very realistic and easy to understand especially for industrial personnel/technicians. There are other measures of closed-loop performance such as Huang and Shah (1999)'s LQG performance benchmark but it needs a reliable process model. Tyler and Morari (1996)'s method based on a maximum likelihood theory is considered a complicated method (Kozub 1996, Harris *et al.* 1999). Ko and Edgar (2000) developed an algorithm for the estimation of MV performance that does not require knowledge of the interactor matrix. However, it does require the first few Markov parameters of the process and closed-loop data from the process.

Minimum Variance (MV) control gives the best theoretically possible control performance and provides information on the basic performance limitations due to time delay in the closed-loop system. Although MV control is not recommended for practical process applications, it can be used as the first level performance assessment.

Bitmead *et al.* (1990) showed that in limiting cases, LQG and GPC (or MPC) are equivalent since both controllers are designed based on similar performance objectives i.e. the GPC control law can be expressed as a receding horizon LQG control law. Since LQG and MPC have similar control structures, it is reasonable to use LQG performance as the benchmark for MPC performance assessment. Huang and Shah (1999) solved the LQG control problem using the GPC (or MPC) formulation and MATLAB MPC toolbox. Patwardhan (1999) discussed performance assessment of MPC using the design case as a benchmark.

In this chapter, performance assessment procedures based on a time series model and a minimum variance performance criterion are briefly reviewed in Sections 10.2-10.3. Section 10.4 discusses performance assessment using closed-loop ARMarkov identification and compares the advantages/disadvantages of this method with Huang *et al.* (1997*a*)'s FCOR algorithm and Kozub (1996)'s method using time series models. In Section 10.5 performance assessment of ARM-MPC controllers using LQG as a benchmark is discussed. Section 10.6 then extends this work to include robustness analysis and conditions for robust stability.

10.1.2 Robustness Analysis

In reality, plant models are only an approximation of the true plant. In most applications, lower order models are used to keep the controller design simple and hence

modelling error or Model Plant Mismatch (MPM) is always present. Therefore, when designing a controller, it is wise to ensure that the closed-loop system is robust to model uncertainties and disturbances. Since MPC is designed based on a plant model and MPM is always present, it is necessary to design an MPC that is robust to MPM and to monitor the robustness of the designed MPC under closed loop conditions.

Robustness analysis of model predictive controller started in the late seventies (e.g. Richalet *et al.* (1978)). Research has continued and in the last couple of decades several papers have been published on the robustness of MPC. Some well known books (e.g. (Morari and Zafiriou 1989, Skogestad and Postlethwaite 1996)) also discuss this issue. Campo and Morari (1986)used an H_∞ formulation of MPC to attain robustness. Genceli and Nikolaou (1993),Rawlings and Muske (1993), Hrissagis *et al.* (1995), Nicolao *et al.* (1996)all presented significant research on the robustness of MPC systems including some work in the non-linear area (Meadows *et al.* 1995*a*, Scokaert *et al.* 1997). Kothare *et al.* (1996)discussed MPC robustness using linear matrix inequalities. There are a number of review papers (e.g. (Garcia *et al.* 1989, Froisy 1994, Rawlings *et al.* 1994, Lee 1996, Qin and Badgwell 1996)), in which significant work in this area is discussed and compared. Morari and Lee (1999), Qin and Badgwell (1996)discussed possible future directions for research in MPC robustness. Banerjee (1996) discussed the robustness of long range predictive control (LRPC). More recently Mayne *et al.* (2000) reviewed the developments in the robust MPC area.

The objective of this thesis was not to develop new robustness theory for MPC, but to apply existing tools for the robustness analysis to ARM-MPC's and to show the advantages of using the ARM-MPC structure. Robustness analysis of ARM-MPC is discussed in Section 10.6. Section 10.6.1 shows how the closed-loop ARMarkov method can be used with the classical Small Gain Theorem (SGT) for robustness analysis and Section 10.6.2 analyzes the robustness of ARM-MPC using matrix perturbation methods.

10.1.3 Ultimate Objective

All MPC users would like *performance* that approaches the best possible and *robustness* that would handle all possible system uncertainties and changes. Unfortunately,

this is theoretically impossible since in all practical cases an increase in performance is accompanied by a decrease in robustness, and vice versa. Therefore, considerable effort was put into developing a *practical benchmark* for evaluating the designed MPC and *effective tuning methods* that could be used *on-line* to adjust the performance/robustness trade-off, e.g. when system uncertainty and disturbances were known to be small the system could be tuned for higher performance (since, by definition, the need for robustness is lower). Alternatively, during periods of process upset (or poor model identification and output prediction) the robustness could be increased (at the cost of poorer performance). In a very real sense, this was the overall goal of the thesis. All of the proceeding theoretical developments and system formulations can be regarded as simply necessary prerequisites to this ultimate goal as discussed in Chapters 10 and 11.

10.2 Performance Assessment Using a Minimum Variance Benchmark

Minimum Variance (MV) control is, theoretically, the best possible control but it is impractical in industrial process control applications because of its demand for excessive control action and poor robustness. However, it is important because it defines the best possible control and provides a global lower bound on the process variance so that the user can evaluate whether it is worth trying to improve the current performance of the application. The basic performance limitation of a controller is due to the time delay of the system because the process output variance is controller invariant upto d time steps where d is the process time delay. To solve the MV control problem, it is necessary to know the time delay or interactor matrix. The interactor matrix for MIMO systems is a generalization of time delay in SISO systems. The MV performance benchmark has been used in many industrial applications mainly because it requires information only on time delay/interactor matrix and does not require the complete process model. Interactor matrix estimation is therefore a crucial step in MV benchmarking. There are different types of interactor matrices and different methods for estimating it. Huang and Shah (1999) showed the advantages of the unitary interactor matrix and described a procedure to estimate it

from process Markov parameters. They also proposed an algorithm called the Filtering and CORrelation (FCOR) algorithm, to obtain an MV benchmark using the unitary interactor matrix and correlation analysis. Kamrunnahar *et al.* (2000) (also described in Chapter 4) used the approach of Huang and Shah (1999) for the factorization of unitary interactor matrix obtained using the Markov parameters produced by the ARMarkov-Least Squares identification method described in Chapter 2. In the next subsection the FCOR algorithm is briefly reviewed and later compared with the ARMarkov-based method.

10.2.1 Filtering and CORrelation (FCOR) Algorithm

The FCOR algorithm, proposed by Huang and Shah (1999), assumes that the unitary interactor matrix has been estimated using open-loop, or closed-loop identification or any other method. The basic steps are:

- filter the interactor-free output response, $\widetilde{y}_t$ in Figure 10.1
- whiten $\widetilde{y}_t$ to get a white noise estimate, $\hat{a}_t$
- use correlation analysis to estimate correlation coefficients, $\rho_{\widetilde{y}\hat{a}}$.
- calculate the Minimum Variance benchmark

The FCOR method uses routine operating data and is simple in nature. However, two different output (or output error) characteristics (settling time, overshoot, amount of cycling etc.) can result in the same correlation coefficients and hence the results are insufficient if one is interested in closed-loop transient response rather than only variance. The correlation is between whitened noise a_t and the output y_t. There is no extra disturbances affecting the correlation analysis.

10.3 User-specified Benchmarks for Closed-loop Performance Analysis

Kozub (1996) pointed out the advantages of using time series modeling for MV benchmarks as well as performance assessment against a desired closed-loop settling time, time constant, overshoot etc. To use this method, the closed-loop system must be

excited using a dither signal. This method also needs a priori information on the delay structure and/or interactor matrix estimation. The main steps in this method are:

- time series modeling or parametric ARMA modeling of the output (or output error)
- computing an MV benchmark
- pulse response data generated from the identified ARMA model gives useful information regarding closed-loop settling time, overshoot, amount of cycling and closeness to MV control.

This method uses routine operating data but the parametric ARMA modeling requires careful model order selection and the parameters are affected by colored disturbances.

10.4 Performance Analysis Using Closed-loop AR-Markov Identification

Kamrunnahar *et al.* (2000) have shown that use of the Markov parameters obtained using ARMarkov identification (discussed in Chapter 2) leads to more accurate interactor matrix estimation. The ARMarkov model is a combination of parametric and non-parametric models and contains the Markov parameters explicitly. Due to the model structure, the ARMarkov method gives a consistent estimate of the Markov parameters and in turn a better estimate of the unitary interactor matrix even in the presence of colored disturbances (Kamrunnahar *et al.* 2000). In this section a closed-loop (CL) ARMarkov method for performance assessment is developed. *It is shown that without exact specification of the delay structure or the model order, the proposed method gives a reliable MV benchmark (unitary interactor matrix), produces realistic performance measures in terms of closed-loop settling time, overshoot etc. without additional computation. It also generates a transfer function model that can be used as a measure of closed-loop performance as well as robustness and for on-line tuning of the controller. Since the proposed method uses simple linear regression, the*

computational cost is reasonable. The following subsection explains how the CL AR-Markov method gives different measures of controller performance as well as how it can be used for controller tuning.

10.4.1 The ARMarkov Identification

The ARMarkov time-domain model of a discrete, linear, time invariant process is a combination of a non-parametric and a parametric model and is defined as

$$\begin{aligned} y(k) &= -\sum_{j=1}^{n} \alpha_{\mu,j} y(k-\mu-j+1) + \sum_{j=1}^{\mu} h_{j-2} u(k-j+1) + \sum_{j=1}^{n} \beta_{\mu,j} u(k-\mu-j+1) \\ &\quad +\varepsilon(k) \end{aligned} \tag{10.1}$$

The ARMarkov representation, with μ Markov parameters, corresponding to the transfer function, $G(z)$ is given by

$$G(z) = \frac{h_0 z^{\mu+n-1} + \cdots + h_\mu z^n + \beta_{\mu,1} z^{n-1} + \cdots + \beta_{\mu,n}}{z^{\mu+n-1} + \alpha_{\mu,1} z^{n-1} + \cdots + \alpha_{\mu,n}} \tag{10.2}$$

Note that it involves only the first μ Markov parameters $h_{0,\cdots}, h_{\mu-1}$. The parameters $\alpha_{\mu,1}, \cdots, \alpha_{\mu,n} \in R^{1\times n}$ are functions of the ARX coefficients and the Markov parameters (Akers and Bernstein 1997). Here n is the order of the ARMarkov model.

The estimated parameter vectors using the ARMarkov method are

$$\widehat{\mathbf{W}}_\mu = \left[-\widehat{\mathbf{A}}_\mu \quad h_0 \quad \cdots \quad h_{\mu-1} \quad \widehat{\mathbf{B}}_\mu\right] \tag{10.3}$$

$$\widehat{\mathbf{A}}_\mu = [\alpha_1, \cdots, \alpha_n] \in R^{1\times n} \tag{10.4}$$

$$\widehat{\mathbf{B}}_\mu = [\beta_1 \ \cdots \ \beta_n] \in R^{1\times n} \tag{10.5}$$

The Markov parameters $h_{0,\cdots}, h_{\mu-1}$ estimated by the ARMarkov method have better statistical properties than the Markov parameters estimated by other linear regression methods such as FIR, ARX as discussed by Kamrunnahar *et al.* (2000)and also in Chapter 4.

Closed-loop ARMarkov Method

A simple closed-loop system is shown in Figure 10.1.

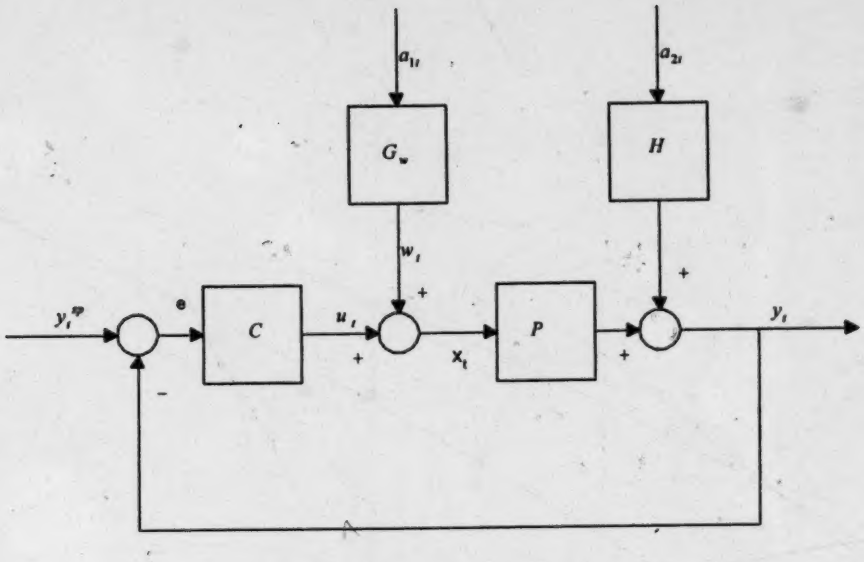


Figure 10.1: Closed-loop system

For this closed-loop system with $y_t^{sp} = 0$,

$$y_t = PSw_t + HSa_t \tag{10.6}$$

$$\text{and } x_t = Sw_t - SCHa_t \tag{10.7}$$

and when a dither signal is inserted at the set-point i.e. $y_t^{sp} \neq 0$, $w_t = 0$

$$y_t = PCSy_t^{sp} + HSa_t \tag{10.8}$$

$$x_t = CSy_t^{sp} - HCSa_t \tag{10.9}$$

$$\text{and } e_t = Sy_t^{sp} - HSa_t \tag{10.10}$$

When the signals y_t and w_t are used as the outputs and inputs respectively for the ARMarkov identification, the estimated parameter vector $\widehat{W}$ includes the Markov parameters h_0 through $h_{\mu-1}$ of the transfer function model PS where P is the process transfer function and S is the sensitivity function. When e_t and y_t^{sp} are used, the Markov parameters and the model for the sensitivity function, S of the closed-loop system are estimated directly.

Huang and Shah (1999) showed that the interactor matrix is feedback invariant, i.e. the interactor matrix of the open-loop system is the same as the interactor matrix

of the closed-loop system. As stated in Huang and Shah (1999), *Identification of the interactor matrix under CL conditions provides a more realistic estimate than under open-loop conditions in the sense that it gives the interactor matrix of the process under the current operating point. · · · . Computationally, a direct identification of the first few Markov parameters is also more desirable than an identification of the full transfer function matrix first, followed by its transfer to Markov parameters. Therefore, the factorization of the interactor matrix from the first few Markov parameters is preferred to the factorization of the interactor matrix from the transfer function matrix.* The CL ARMarkov method estimates the Markov parameters *directly* rather than estimating the transfer function first and then using it to generate the Markov parameters. Kamrunnahar *et al.* (2000) showed that the Markov parameters obtained using CL ARMarkov identification give a more accurate unitary interactor matrix than the unitary interactor matrix calculated from Markov parameters from other identified linear regression models. The reason behind the improved accuracy lies in the direct estimation of the required μ Markov parameters and the consistency of the Markov parameters even in the presence of non-white disturbances. For more information on the calculation of the interactor matrix using CL data, see (Huang and Shah 1999). Unlike correlation analysis or time series modelling, the CL ARMarkov method uses a dither signal as well as the process output response during the identification procedure. Additional information obtained using the parameters from the same ARMarkov identification are described below.

Remark 10.1 *When enough Markov parameters are estimated, the ARMarkov approach gives direct CL performance measures in terms of settling time, overshoot etc. The achieved CL performance can be compared with desired CL response specifications.*

Remark 10.2 *Typically the controller being used in the closed-loop system is known. Therefore, the sensitivity function and/or the complementary sensitivity function can be determined using the following procedure.*

The transfer function, PS is estimated using the ARMarkov method and the closed-loop signals w_t and y_t. For a known controller, C, the complementary sensitivity function for a SISO system is:

$$T = PCS = \frac{PC}{1+PC} \tag{10.11}$$

and the sensitivity function is:

$$S = 1 - T = \frac{1}{1+PC} \tag{10.12}$$

The sensitivity and the complementary sensitivity functions are obtained without much additional computation and can be used as measures of closed-loop performance. The identification method uses simple linear regression. However, the order of the model is high due to the overparameterized structure of the model.

The two transfer functions, T and S can also be used for robustness analysis of the system as discussed in section 10.6.1.

Remark 10.3 *When a dither signal is inserted at the set-point, the signals e_t and y_t^{sp} in Figure 10.1 are used in the ARMarkov identification and the Markov parameters and the model for the sensitivity function, S are estimated. S is a measure of closed-loop performance e.g. it determines how quickly the output error, e_t converges to zero i.e. the output, y_t follows the set-point, y_t^{sp}. S can also be used for robust stability, and robust performance analysis. As mentioned above, the CL time constant, settling time etc. can be measured directly using the estimated Markov parameters of the sensitivity function.*

Two-step Closed-loop Identification

Closed-loop identification of a process model has been an attractive field of research for the last couple of decades for many reasons e.g. under CL conditions, the model parameters are estimated under the nominal operating conditions used for a linearized model. Van Den Hof and Schrama (1993) used a two step method called w-filtering for CL identification where the sensitivity function is estimated in the first step and the input signal, w_t is filtered in the second step to estimate the plant model. Huang (1997) proposed a two-step closed-loop identification procedure called y-filtering that filters the output signal, y in the second step. In this subsection the y-filtering method is briefly reviewed and then it is shown how the CL ARMarkov identification can be used for the two step identification.

The two-step ID procedure posses the CL identification into the following two single, open-loop ID problems

1. x_t and w_t are used to estimate the sensitivity function, S using the relationship in (10.7) with $y_t^{sp} = 0$.

2. output y_t and input w_t are filtered by the estimate of S so that using (10.6)

$$y_t^f = \frac{y_t}{\widehat{S}} = Pw_t + \frac{H}{\widehat{S}}a_{2t}$$

where $\widehat{S}$ is the estimate of S. Then using y_t^f and w_t as the output and input, the model of the plant P is estimated. For more details on this procedure and the convergence of $\widehat{S}$ to S, see (Huang and Shah 1999).

Remark 10.4 *The Markov parameters which are directly available from the estimated ARMarkov model of the plant can be used for the estimation of interactor matrix and in turn, the MV performance benchmark.*

For the closed-loop system in Figure 10.1 with $y_t^{sp} \neq 0$, $w_t = 0$, the model CS is estimated using the relationship in (10.9) as the first step in the two-step ID. In the second step, the Markov parameters and the model of the plant P are estimated using signals $\frac{y_t}{CS}$ and y_t^{sp} as described by equation (10.8). The results can be used for performance analysis as discussed in Remark 10.4.

Remark 10.5 *In summary, the ARMarkov approach provides multiple performance metrics as well as the plant model. The advantage of the ARMarkov method over other methods is that it does not require a priori specification of the delay structure and when combined with disturbance model estimation (e.g. through filtering as in the prediction error method), it provides a reliable model identification procedure. On-line model updating can be done using recursive parameter estimation which saves significant computational cost due to the overparameterization in the ARMarkov model.*

In Table 10.1, the properties of the closed-loop ARMarkov identification are compared with the properties of the FCOR method of Huang (1997) and the time series model of Kozub (1996).

Table 10.1: Comparison of the Advantages of Different Performance Assessment Methods

Method	FCOR	Time Series	CL ARMarkov
data	output(error)	output(error)	output+dither signal
time delay	a priori spec. of Markov param.*	a priori spec. of Markov param.	gives Markov param., no a priori spec. reqd
MV benchmark	available	available	available
statistical properties of Markov param.	-	-	consistent, low variance
CL settling time, overshoot	-	from pulse response	directly available**
sensitivity function	-	noise model + filtering	directly available
model ID	-	-	1st step in two step CL ID
model order spec.	-	needs careful spec.	approx. order

*param.=parameters

** if μ, the number of estimated Markov parameters is large enough.

Example 10.1

Consider the third order process

$$P(s) = \frac{e^{-2s}}{(s+1)(3s+1)(5s+1)} \tag{10.13}$$

the discrete transfer function of which, with sampling time $T_s = 1$, is

$$P(z) = \frac{0.0077z^{-3} + 0.0212z^{-4} + 0.0036z^{-5}}{1 - 1.9031z^{-1} + 1.1514z^{-2} - 0.2158z^{-3}} \tag{10.14}$$

Let

$$C(z) = 1$$

For this process, the ARMarkov identification method is used with closed-loop signals w_t and y_t as the inputs and outputs respectively. The estimated impulse response plot for the transfer function, PS is shown in Figure 10.2. The time delay for the CL system is the same as the open-loop time delay since the time delay is feedback invariant. Interactor matrix estimation for MIMO systems using the CL ARMarkov method are discussed in Chapter 5. However, to simplify the interpretation of the

following results, only SISO systems are considered here. Using the controller information, the Bode magnitude plots for the estimated sensitivity and complementary sensitivity functions are shown in Figures 10.3(a) and (b) respectively. The intention here was not to design the controller, but to show how the CL ARMarkov method can be used for performance analysis.

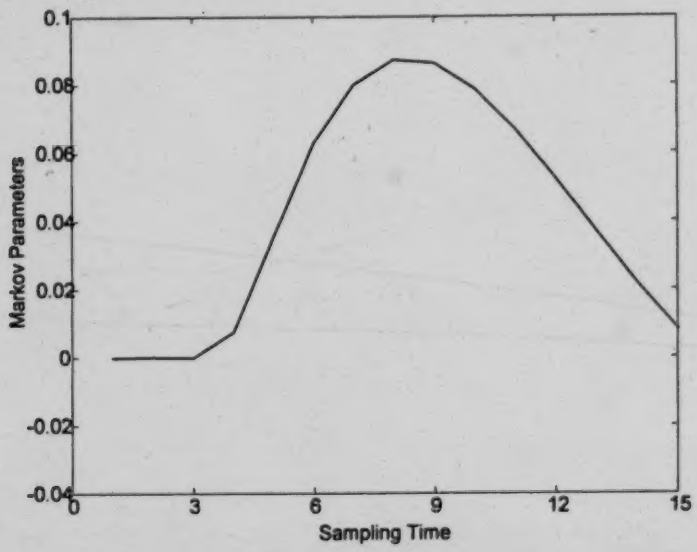


Figure 10.2: Markov parameters using CL ARMarkov method.

Example 10.2

For the same process in (10.13), closed-loop signals w_t and x_t are used in the AR-Markov method to estimate the sensitivity function in the first step and following the two step identification procedure outlined in subsection 10.4.1, the Markov parameters and the plant model are estimated in the second step. The estimated Markov parameters are compared with the actual Markov parameters in Figure 10.4(a) and the estimated sensitivity function is plotted in Figure 10.4(b). The estimated Markov parameters are essentially the same as the actual ones.

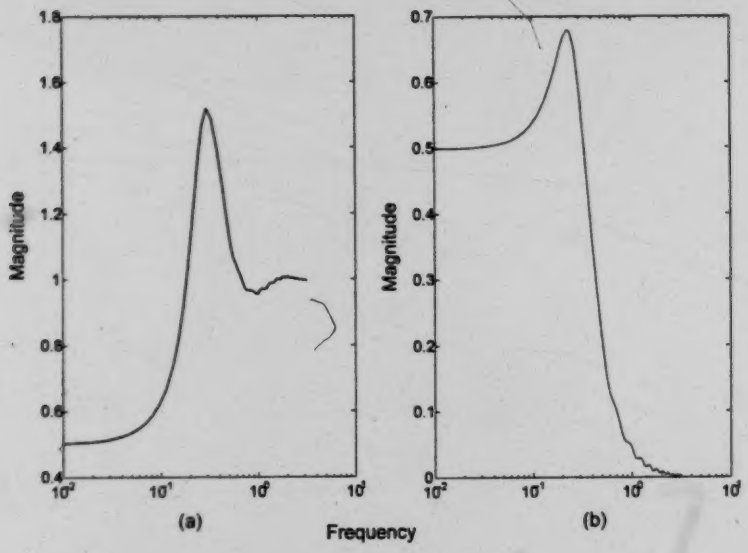


Figure 10.3: (a) Sensitivity and (b) complementary sensitivity functions using the CL ARMarkov method.

10.5 Performance Analysis Using Linear Quadratic Gaussian (LQG) Control as the Benchmark

Although minimum variance (MV) control provides the global lower bound on the output variance, it may not be possible to achieve this lower bound in practice because the MV controller objective does not include any penalty on the input variance and hence, typically requires very large control effort. It is much more practical and desirable to assess the performance of the existing controller assuming some restrictions on the control move.

A Linear Quadratic Gaussian (LQG) controller that imposes a penalty on the control move provides the best *achievable* control performance and provides a basis for determining whether there is any potential to improve the performance of the existing controller. The objective functions of LQG and model-based predictive controllers (MPC) are similar as noted below.

The solution of an MPC control problem is obtained by minimizing a cost function

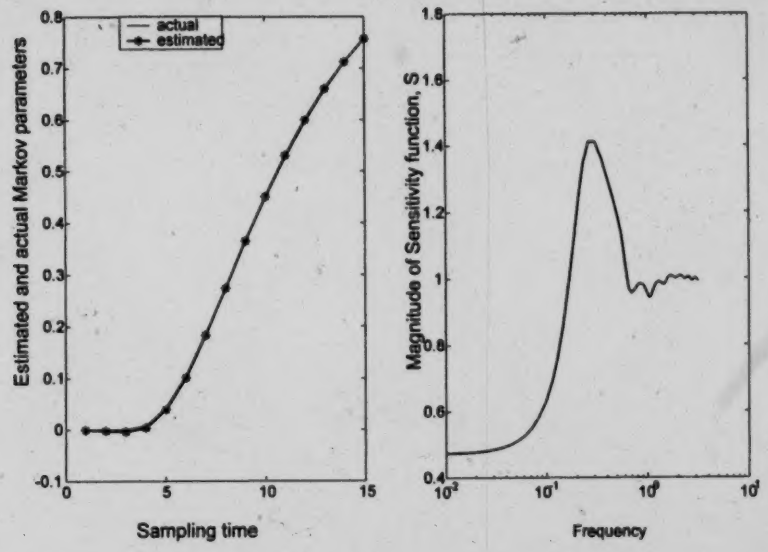


Figure 10.4: (a) Markov parameters, actual and estimated, (b) estimated sensitivity function.

such as,

$$J = E\left\{\sum_{j=1}^{N_2}(y_{t+j} - r_{t+j})^T Q_j (y_{t+j} - r_{t+j}) + \lambda \sum_{j=0}^{N_u} \Delta u_{t+j}^T R_j \Delta u_{t+j}\right\} \qquad (10.15)$$

where N_2 and N_u are the prediction horizon and control horizon respectively. The cost function used in the solution of the LQG tracking problem can be written as

$$J = E\left\{(y_t - r_t)^T Q_{lq} (y_t - r_t) + \Delta u_t^T R_{lq} \Delta u_t\right\} \qquad (10.16)$$

As $N_u = N_2 \to \infty$, the above two performance criteria converge to each other. For the solution of the LQG regulator problem, the LQG tracking problem can be recast into a regulator problem. Bitmead *et al.* (1990) showed that GPC (or equivalently MPC) can be embedded into the LQG receding horizon principle and can be solved as a special case of LQG. Therefore, it is reasonable to assess the MPC controller performance against the best achievable LQG performance. It is beyond the scope of this thesis to discuss all the relevant LQG theory and properties. However, the solution of the LQG is briefly reviewed below.

Consider a state space system

$$\begin{aligned} x(k+1) &= A_x x(k) + B_x u(k) + G_x w(k) \\ y(k) &= C_x x(k) + z(k) \end{aligned}$$

The solution of the state feedback control law is expressed as

$$u(k) = -K\widehat{x}(k) \tag{10.17}$$

where K is the state feedback gain that is calculated as

$$K = A_x P_x C_x^T \left(R_2 + C_x P_x C_x^T\right)^{-1} \tag{10.18}$$

where P_x is the steady state solution of the Riccati equation

$$P_x(k+1) = A_x P_x(k) A_x^T + R_1 - A_x P_x(k) C_x^T \left(R_2 + C_x P_x C_x^T\right)^{-1} C_x P_x(k) A_x^T \tag{10.19}$$

R_1, R_2 are the covariance matrices of $w(k)$ and $z(k)$ respectively.

The feedback observer or state estimator can be written as

$$\widehat{x}(k|k) = \widehat{x}(k|k-1) + K^f \left(y(k) - C_x x(k|k-1)\right) \tag{10.20}$$

$$\widehat{x}(k+1|k) = A_x \widehat{x}(k|k) + B_x u(k) \tag{10.21}$$

where K^f is the Kalman filter gain that can be calculated by solving another Riccatti equation (Bitmead *et al.* 1990). Huang and Shah (1999)discussed the LQG solution via GPC or MPC. When $N_u, N_2 \to \infty$, the GPC solution converges to the LQG solution. However, in practice, after a large but finite value of N_u, N_2, the solution of LQG is obtained as the solution of MPC. Chen and Francis (1995)and Patwardhan (1999)discussed the LQG solution, for uncorrelated state and measurement noise, using the solution of a Lyapunov equation.

LQG has been used as a benchmark/metric for performance assessment by researchers such as Huang and Shah (1999). When the LQG objective function in (10.16) is modified to reflect the control weighting of the MPC objective function (10.15) i.e. $R_{lq} = \lambda R$, different control laws are obtained by solving the LQG problem for different numerical values of λ. For each value of λ, the output variance can be plotted against the input variance to obtain a performance trade-off curve as

shown in Figure 10.5. This trade-off curve represents the best achievable LQG control performance. Then the existing controller performance in terms of input/output variances can be assessed with respect to the trade-off curve. The distance of the existing control performance from the trade-off curve is a measure of the achieved performance and in the design of C_0 and C_1 used with $\alpha-$tuning discussed in the next chapter.

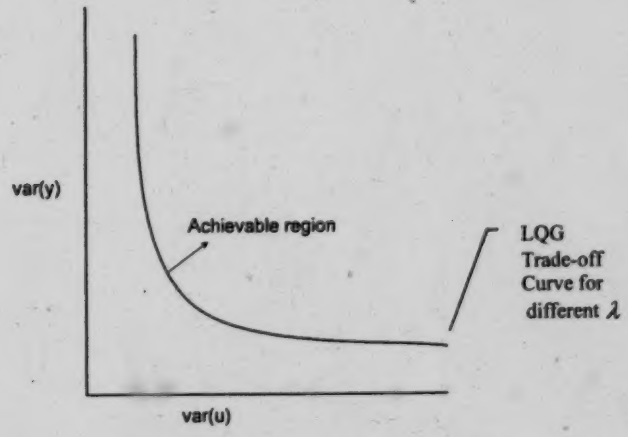


Figure 10.5: LQG performance trade-off curve

Example 10.3

Consider the third order process in (10.13). An ARM-MPC was designed for this process in Chapter 8. An LQG was designed for this process using the estimated ARMarkov model in the state space form given in (8.30). The LQG trade-off curve is shown in Figure 10.6. The tracking performance of the ARM-MPC is assessed against this LQG curve. The effect of control horizon M and prediction horizon N_2 are shown in Figures 10.6 and 10.7 respectively. It is difficult to say which controller is better because the performance (aggressive/conservative) requirement is application dependent. However, the performance of the controller can be easily assessed relative to the LQG benchmark.

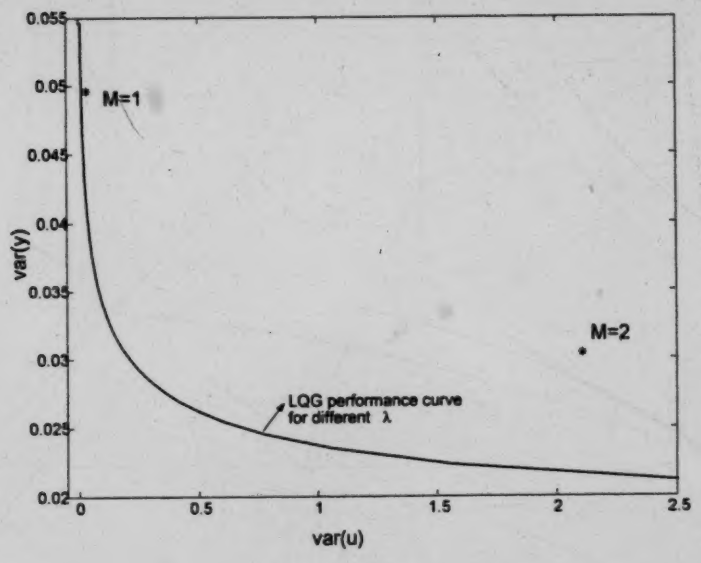


Figure 10.6: Effect of control horizon on the performance of the ARM-MPC.

10.6 Robustness Analysis of ARM-MPC

Every controller should be designed such that it is insensitive to model plant mismatch (MPM) and the ARM-MPC should not be any exception to that. As defined by many textbook authors (e.g. (Morari and Zafiriou 1989, Skogestad and Postlethwaite 1996, Doyle *et al.* 1992)), MPM can be placed into two main categories:

1. **Structured uncertainty**. This type of uncertainty is very specific because the model structure is assumed to be the same as the process but the model parameters are uncertain. Gain mismatch, time constant and/or time delay mismatch are the most common examples of structured uncertainty.

2. **Unstructured uncertainty**. In many cases, the structure of the actual process is unknown or is approximated by lower order models. One of the advantages of the unstructured or disk-like uncertainty is that it includes the unmodelled dynamics of the process. Moreover, analysis is simpler using this uncertainty. Multiplicative, additive and feedback are the most common unstructured uncertainties. Some of these uncertainties are illustrated in Figures 10.8-10.10.

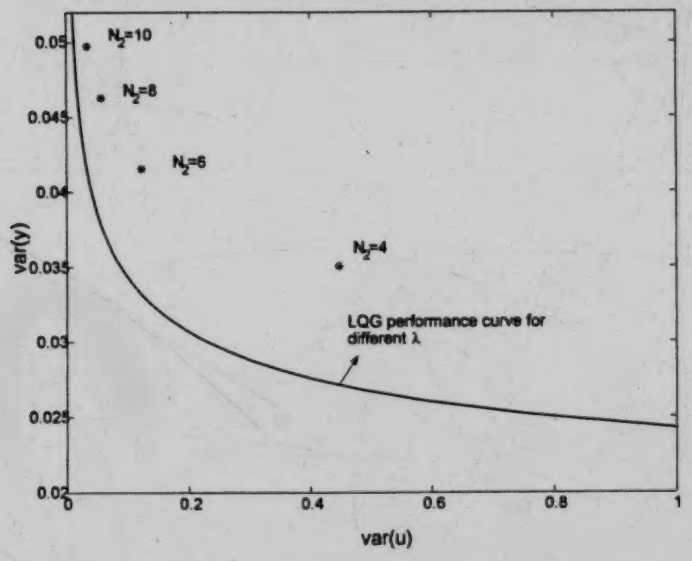


Figure 10.7: Effect of prediction horizon, N_2 on the performance of the ARM-MPC.

The best uncertainty type to be used for robustness analysis is application dependent. Structured or parametric uncertainties can also be expressed in terms of unstructured uncertainty.

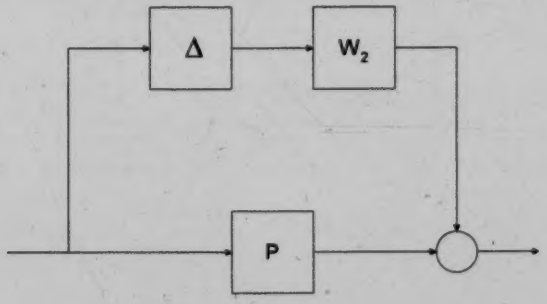


Figure 10.8: Multiplicative uncertainty

The weighting function W_2 is a fixed, stable transfer function and Δ is a variable stable transfer function such that $\|\Delta\|_\infty \leq 1$. The robust stability of the controller is analyzed either in the time domain (e.g. by solving Lyapunov equations) or in the frequency domain. Frequency domain analysis is used in this thesis. The Small Gain

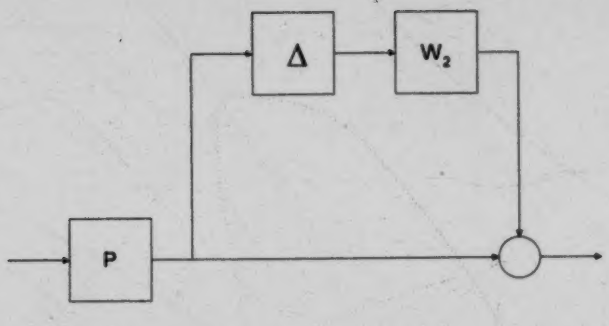


Figure 10.9: Additive uncertainty

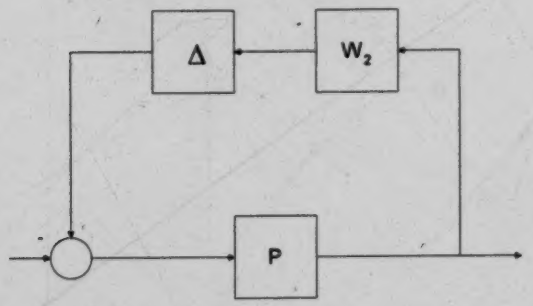


Figure 10.10: Feedback uncertainty

Theorem (SGT) is a very well known robustness analysis criterion that provides a sufficient condition for robust stability of a given system. However, the price for the sufficiency lies in the conservativeness of the approach.

Uncertainty estimation is not the topic covered here. For the purpose of robustness analysis, model uncertainties can be estimated by following methods available in the literature. Banerjee (1996)used signal processing to estimate the uncertainties. Patwardhan (1999)presented an approach to determine model uncertainty from closed-loop data.

Theorem 2 *Small Gain Theorem (SGT)(Skogestad and Postlethwaite 1996): Consider a system with a stable loop transfer function, $L = PC$. The closed-loop system is stable if*

$$\|L\| \leq 1 \tag{10.22}$$

where $\|\cdot\|$ denotes any matrix norm.

For a closed-loop system with the multiplicative uncertainty, the special form of the SGT is

Theorem 3 *The closed-loop system is robustly stable if*

$$\|W_2 T\|_\infty \leq 1 \tag{10.23}$$

The SGT will produce different stability conditions for different types of uncertainties.

10.6.1 Closed-loop ARMarkov model and the Small Gain Theorem (SGT)

Feedback uncertainty

The family of plant models for the feedback uncertainty shown in Figure 10.10 is written as

$$\mathcal{P} = \frac{P}{(1 + \Delta W_2 P)} \tag{10.24}$$

where P is the nominal model and $\mathcal{P}$ is the perturbed plant. The following example shows how parametric uncertainty is expressed in terms of the feedback uncertainty.

Example 10.4 *(Doyle* et al. *1992)Consider the family of plant transfer functions*

$$\frac{1}{s^2 + as + 1}, \qquad 0.4 \leq a \leq 0.8 \tag{10.25}$$

$$a = 0.6 + 0.2\Delta, \qquad -1 \leq \Delta \leq 1 \tag{10.26}$$

This family of transfer functions can be expressed as

$$\frac{P(s)}{1 + \Delta W_2(s) P(s)}, \qquad -1 \leq \Delta \leq 1 \tag{10.27}$$

where

$$P(s) = \frac{1}{s^2 + 0.6s + 1}, \qquad W_2(s) = 0.2s$$

The robust stability criteria for this uncertainty can be written as

$$\|W_2 PS\|_\infty < 1$$

Using closed-loop input-output data, *the ARMarkov method directly estimates the transfer function, PS* as illustrated by the performance analysis in Section 10.5. The same transfer function can be used for robustness analysis using the robust stability condition for feedback uncertainty presented above.

Multiplicative and Additive uncertainty

Other common and simple forms of uncertainties include multiplicative and additive uncertainties which are illustrated by Figures 10.8-10.9. Gain and/or time delay uncertainties can be embedded into the multiplicative uncertainty as follows.

Example 10.5 *Consider the nominal transfer function*

$$P(s) = \frac{1}{\tau s + 1} \tag{10.28}$$

with the family of perturbed transfer functions

$$\mathcal{P}(s) = e^{-\tau_1 s}\frac{1}{\tau s + 1}, \qquad 0 \le \tau_1 \le 0.1 \tag{10.29}$$

$$= (1 + \Delta W_2)\, P, \qquad \|\Delta\|_\infty < 1 \tag{10.30}$$

The robust stability conditions for the multiplicative and additive uncertainties are

$$\|W_2 T\|_\infty < 1, \text{for multipicative perturbation model } \mathcal{P}(s) = (1 + \Delta W_2)P \tag{10.31}$$

$$\|W_2 CS\|_\infty < 1, \text{for additive perturbation model } \mathcal{P}(s) = P + \Delta W_2 \tag{10.32}$$

where S and T are the sensitivity and complementary sensitivity functions respectively for the closed-loop system.

As discussed in Section 10.5, for a known controller, *the closed-loop transfer functions, CS and T are produced directly by the CL ARMarkov identification.* Hence, robust stability can be determined for either of the two uncertainty models using the models produced by the ARMarkov approach and closed-loop data.

Robust Performance Analysis

Closed-loop performance and robust stability analysis have been discussed separately in Sections 10.5 and 10.6.1 respectively using the same CL ARMarkov model. *Robust performance*, which combines both performance and robust stability, is much more important from a practical point of view. The condition for robust performance, using multiplicative uncertainty, is (e.g. (Doyle *et al.* 1992))

$$\| |W_1 S| + |W_2 T| \|_\infty < 1$$

Therefore, the robust performance of the closed-loop system can be analyzed using the CL models T and S estimated by the ARMarkov method and choosing appropriate weighting functions $W_1$and W_2.

10.6.2 Matrix Perturbation Method

The systems and control literature is rich in well established and simple methods of stability analysis in the state space domain. Perturbations or uncertainties in the system and/or the input matrices are considered for robust design and/or analysis of the closed-loop system. Closed-loop pole locations, represented by the eigenvalues of the closed-loop system matrix, indicate the robust stability margin or stability bounds of the system. Typically parametric uncertainties in terms of matrix perturbations are posed in the additive perturbation form and the small gain theorem (SGT) is applied for the analysis of the robustness. Yedavalli (1985)discussed perturbation bounds for robust stability in the time domain and used a Lyapunov approach for the analysis. Both structured and unstructured perturbations were considered. Dickman (1987)discussed robustness using matrix perturbation theory where perturbations in the system matrix for a continuous time system was considered. Robust stability for discrete time systems using the matrix perturbation method was considered in Qu and Dorsey (1990). Qi (1997)developed robust stability conditions for his dual-model predictive controller (DMPC).

The objective here is to apply the well established theories, such as those mentioned above, for the analysis of the designed ARM-MPC formulated in the (special, dual-model) state space form presented in Chapter 8. The ARMarkov method estimates a time-domain parametric model but an equivalent state space form of the model and MPC formulation were developed to facilitate design and analysis. Since the ARMarkov model is an overparameterized model, there is less possibility of a bias error in the estimated parameters. The system matrix in the ARMarkov state space model includes the ARX equivalent parameters which can be considered as the slow dynamics of the process and the input matrix includes the Markov parameters which can be considered as fast dynamics. The time delay of the system is defined by the first few Markov parameters of the model. The closed-loop system matrix was developed in Chapter 8 for the observer based MPC design.

Consider a discrete time closed-loop system

$$X_{cl}(k+1) = A_{cl}X(k) \tag{10.33}$$

and the perturbed system

$$X_{cl}(k+1) = \bar{A}_{cl}X(k) \tag{10.34}$$

where, using additive uncertainty principle,

$$\bar{A}_{cl} = A_{cl} + \Delta A_{cl}$$

Assume that the nominal system in (10.33) is stable i.e. all the eigenvalues of A_{cl} lie inside the unit circle. In other words, $(e^{jw_i}I - A_{cl})$ is invertible for all frequencies, w. The system in (10.34) is said to be *robust* if all the eigenvalues of $\bar{A}_{cl}$ also lie inside the unit circle.

The family of plants having the closed-loop system matrix, $\bar{A}_{cl}$ is *robustly stable* if $(A_{cl} + \Delta A_{cl} - e^{jw_i}I)$ is nonsingular for all w, i.e. $(A_{cl} + \Delta A_{cl} - e^{jw_i}I)$ is invertible. The sufficient condition for invertibility is (Qu and Dorsey 1990)

$$\left\|(A_{cl} - e^{jw_i}I)^{-1}(\Delta A_{cl})\right\| < 1 \tag{10.35}$$

Since

$$\left\|(A_{cl} - e^{jw_i}I)^{-1}(\Delta A_{cl})\right\| \leq \left\|(A_{cl} - e^{jw_i}I)^{-1}\right\| \|(\Delta A_{cl})\| \tag{10.36}$$

a conservative condition for (10.35) is

$$\left\|(A_{cl} - e^{jw_i}I)^{-1}\right\| \|(\Delta A_{cl})\| < 1 \tag{10.37}$$

$$\left[\underline{\sigma}(A_{cl} - e^{jw_i}I)\right]^{-1} \|(\Delta A_{cl})\| < 1 \tag{10.38}$$

where

$$\left\|(A_{cl} - e^{jw_i}I)^{-1}\right\| = \left[\underline{\sigma}(A_{cl} - e^{jw_i}I)\right]^{-1}$$

The sufficient condition for the perturbed system to be robustly stable can be written as

(10.39)

$$\underline{\sigma}(A_{cl} - e^{jw_i}I) > \overline{\sigma}(\Delta A_{cl}) \tag{10.40}$$

and the robustness bound, ρ_{rb} of the closed-loop system in (10.33) can be expressed as

$$\rho_{rb} = \min_{w} \underline{\sigma}\left(A_{cl} - e^{jw_i} I\right) \tag{10.41}$$

The larger the value of ρ_{rb}, the larger the error for which the perturbed closed-loop system will remain stable.

Theorem 4 *If A_{cl} is a real matrix, then the robustness bound is*

$$\rho_{rb} = \underline{\sigma}\left(A_{cl} - I\right) \tag{10.42}$$

Proof. *see (Qu and Dorsey 1990) for proof.* ■

Robustness Measure of ARM-MPC

Rewrite the closed-loop formulation of the ARM-MPC in (8.44) as

$$\begin{aligned}
\begin{bmatrix} \hat{X}_{arm}(k+1) \\ \overline{X}_{arm}(k+1) \end{bmatrix} &= A_{cl} \begin{bmatrix} \hat{X}_{arm}(k) \\ \overline{X}_{arm}(k-1) \end{bmatrix} \\
A_{cl} &= \begin{bmatrix} (\Phi_{arm} - \theta_{arm} K_{arm-mpc}) & \theta_{arm} K_{arm-mpc} \\ 0 & (I - K_{arm} H_{arm}) \Phi_{arm} \end{bmatrix}
\end{aligned}$$

MPM in this closed-loop system matrix is assumed to be in the open-loop system matrix (slow dynamics), Φ_{arm} and input matrix (fast dynamics), θ_{arm}. Therefore, the perturbed closed-loop system matrix becomes

$$\begin{aligned}
\overline{A}_{cl} &= \begin{bmatrix} a_{cl}^{11} & a_{cl}^{12} \\ a_{cl}^{21} & a_{cl}^{22} \end{bmatrix} \qquad (10.43) \\
a_{cl}^{11} &= (\Phi_{arm} - \theta_{arm} K_{arm-mpc}) + \Delta\Phi_{arm} - \Delta\theta_{arm} K_{arm-mpc} \\
a_{cl}^{12} &= \theta_{arm} K_{arm-mpc} + \Delta\theta_{arm} K_{arm-mpc} \\
a_{cl}^{21} &= (I - K_{arm} H_{arm}) (\Delta\Phi_{arm} - \Delta\theta_{arm} K_{arm-mpc}) \\
a_{cl}^{22} &= (I - K_{arm} H_{arm}) \Phi_{arm} + (I - K_{arm} H_{arm}) \Delta\theta_{arm} K_{arm-mpc}
\end{aligned}$$

with the additive perturbation matrix

$$\Delta A_{cl} = \begin{bmatrix} \Delta a_{cl}^{11} & \Delta a_{cl}^{12} \\ \Delta a_{cl}^{21} & \Delta a_{cl}^{22} \end{bmatrix} \tag{10.44}$$

$$\Delta a_{cl}^{11} = \Delta\Phi_{arm} - \Delta\theta_{arm} K_{arm-mpc}$$

$$\Delta a_{cl}^{12} = \Delta\theta_{arm} K_{arm-mpc}$$

$$\Delta a_{cl}^{21} = (I - K_{arm} H_{arm})(\Delta\Phi_{arm} - \Delta\theta_{arm} K_{arm-mpc})$$

$$\Delta a_{cl}^{22} = (I - K_{arm} H_{arm})\Delta\theta_{arm} K_{arm-mpc}$$

$$\Delta A_{cl} = \mathrm{k_c} \begin{bmatrix} \Delta\Phi_{arm} - \Delta\theta_{arm} K_{arm-mpc} & 0 \\ 0 & \Delta\theta_{arm} K_{arm-mpc} \end{bmatrix} \tag{10.45}$$

$$\Delta A_{cl} = \mathrm{k_c} \Delta A_{cl}^{1} \tag{10.46}$$

$$\mathrm{k_c} = \begin{bmatrix} I & I \\ (I - K_{arm} H_{arm}) & (I - K_{arm} H_{arm}) \end{bmatrix} \tag{10.47}$$

where $\mathrm{k_c}$ is not affected by the perturbations.

When the perturbation matrix, ΔA_{cl} is factored out as $\Delta A_{cl} = \mathrm{k_c}\,(\Delta A_{cl}^{1})$ as shown in (10.45)-(10.47), equation (10.35) can be rewritten for the special structure state space ARM-MPC as

$$\left\| \left(A_{cl} - e^{jw_i} I\right)^{-1} \mathrm{k_c}\,(\Delta A_{cl}^{1}) \right\| < 1 \tag{10.48}$$

Now combining $\mathrm{k_c}$ with $(A_{cl} - e^{jw_i} I)^{-1}$, the sufficient condition for (10.48) can be written as

$$\left\| \left(A_{cl} - e^{jw_i} I\right)^{-1} \mathrm{k_c} \right\| \left\| (\Delta A_{cl}^{1}) \right\| < 1 \tag{10.49}$$

$$\left\| (\Delta A_{cl}^{1}) \right\| < \frac{1}{\left\| \left(A_{cl} - e^{jw_i} I\right)^{-1} \mathrm{k_c} \right\|}$$

$$\left\| (\Delta A_{cl}^{1}) \right\| < \frac{1}{\left\| M(w)\,\mathrm{k_c}(w) \right\|} \tag{10.50}$$

$$\text{where } M(w) = \left(A_{cl} - e^{jw_i} I\right)^{-1} \tag{10.51}$$

Therefore, the robustness bound for the ARM-MPC can be written as

$$\rho_{rb} = \min_{w} \frac{1}{\bar{\sigma}\left[M(w)\,\mathrm{k_c}(w)\right]} \tag{10.52}$$

Remark 10.6 *As shown above, the conservativeness of the typical robustness analysis can be reduced by factoring the known information* $\mathrm{k_c}$ *out of the perturbation matrix and combining it with the known nominal closed-loop matrix.*

Similarly, for other specific types of perturbations, e.g. gain mismatch, delay mismatch, there are other known terms that can be moved from the perturbation matrix to the nominal closed-loop matrix. This is discussed next and illustrated in Table 10.2.

Repeat equations (8.30) through (8.37) for the ARM-MPC

$$\begin{aligned} \hat{X}_{arm}(k) &= \Phi_{arm}\hat{X}_{arm}(k-1) + \theta_{arm}\Delta u(k-1) \\ \hat{y}(k) &= H_{arm}\hat{X}_{arm}(k) \end{aligned}$$

where

$$\begin{aligned} H_{arm} &= \left[\begin{array}{ccccc} 1 & 0 & 0 & \cdots & 0 \end{array}\right]^T_{1\times(\mu+n)} \\ \Phi_{arm} &= \left[\begin{array}{cc} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{array}\right]_{(\mu+n)\times(\mu+n)}, \quad \theta_{arm} = \left[\begin{array}{c} S_1 \\ S_2 \end{array}\right] \end{aligned}$$

$$\begin{aligned} \phi_{11} &= \left[\begin{array}{cc} 0_{\mu\times 1} & I_{\mu\times\mu} \end{array}\right]_{\mu\times(\mu+1)}, \quad \phi_{12} = 0_{\mu\times(n-1)} \\ S_1 &= \left[\begin{array}{cccc} s_1 & s_2 & \cdots & s_\mu \end{array}\right]^T_{\mu\times 1} \end{aligned}$$

For $n = 3$,

$$\phi_{21^\bullet} = \left[\begin{array}{cccc} -(\alpha_2-\alpha_1) & -\alpha_1 & 0_{1\times(\mu-3)} & 1 \\ 1 & 0 & 0_{1\times(\mu-3)} & 0 \\ 0 & \alpha_2 & 0_{1\times(\mu-3)} & 0 \end{array}\right]^T_{n\times(\mu+1)}$$

$$\phi_{22} = \left[\begin{array}{cc} -(\alpha_3-\alpha_2) & 1 \\ 0 & 0 \\ \alpha_3 & 0 \end{array}\right]^T_{n\times(n-1)}$$

and $$S_2 = \left[\begin{array}{ccc} s_{\mu+1}-\alpha_1 s_1 & 0 & \beta_3 \end{array}\right]^T$$

Although the structure of ϕ_{21}, ϕ_{22} and S_2 are different in ARM-MPC than in the equivalent structure of the DMPC of Qi (1997), it is straightforward to show that the MPM structure is the same for both methods but with different coefficients. The details of the derivation are not repeated here, only the final results are given in Table 10.2. As the known information (k_c) in the CL system matrix is factored out, the conservativeness of the robustness bounds is reduced and the part $\Delta\mathbf{A}^1_{cl}$ that is affected by uncertainties is used to get the new, less conservative robustness bounds. This is illustrated in Example 10.6 and in Figure 10.12.

Table 10.2: Factorization of the uncertainty matrix for different MPMs

(a)

	ΔA_{cl}^{1} is defined in (10.45) (It is affected by uncertainties)
General	$\begin{bmatrix} \Delta\Phi_{arm} - \Delta\theta_{arm}K_{arm-mpc} & 0 \\ 0 & \Delta\theta_{arm}K_{arm-mpc} \end{bmatrix}$
For $\Delta\Phi$	$\begin{bmatrix} \Delta\Phi_{arm} & 0 \\ \Delta\Phi_{arm} & 0 \end{bmatrix}$
For $\Delta\theta$	$\begin{bmatrix} -\Delta\theta_{arm}K_{arm-mpc} & 0 \\ 0 & \Delta\theta_{arm}K_{arm-mpc} \end{bmatrix}$
For Δ(gain)	Δk
For Δ(delay)	$\begin{bmatrix} -\Phi_s C_d K_{arm-mpc} & 0 \\ 0 & \Phi_s C_d K_{arm-mpc} \end{bmatrix}$

(b)

	$k_c(w)$ is defined in (10.47) (It is part of CL system matrix that is NOT affected by uncertainties)
General	$\begin{bmatrix} I & I \\ (I - K_{arm}H_{arm}) & (I - K_{arm}H_{arm}) \end{bmatrix}$
For $\Delta\Phi$	I
For $\Delta\theta$	$\begin{bmatrix} I & I \\ -(I - K_{arm}H_{arm}) & (I - K_{arm}H_{arm}) \end{bmatrix}$
For Δ(gain)	$\begin{bmatrix} -\theta_{arm}K_{arm-mpc} & \theta_{arm}K_{arm-mpc} \\ -(I - K_{arm}H_{arm})\,\theta_{arm}K_{arm-mpc} & (I - K_{arm}H_{arm})\,\theta_{arm}K_{arm-mpc} \end{bmatrix}$
For Δ(delay)	$\begin{bmatrix} I & I \\ (I - K_{arm}H_{arm}) & (I - K_{arm}H_{arm}) \end{bmatrix}$

In the dual-model, special structure, state space form of the ARM-MPC the uncertainty is defined by ΔA_{cl} in (10.44). This can be factored into $\Delta A_{cl} = k_c \Delta A^1_{cl}$ as shown in (10.45). For specific forms of uncertainties, e.g. gain or time delay mismatch, ΔA^1_{cl} and k_c can be expressed as shown in Table 10.2. The uncertainties expressed in Table 10.2 lead to less conservative designs (and higher robustness bunds) than can be obtained from general $[A, B, C, D]$ state space methods and robustness theory. Examples of the improvement in robustness bounds are illustrated in Figure 10.12.

Example 10.6

Consider the third order process in (10.13). The robustness bound, ρ_{rb} in (10.52) for the ARM-MPC was determined for this process using the matrix perturbation method and is shown in Figure 10.11. The tuning parameters used are: $M = 1, N_2 = 10, \lambda = 0$.

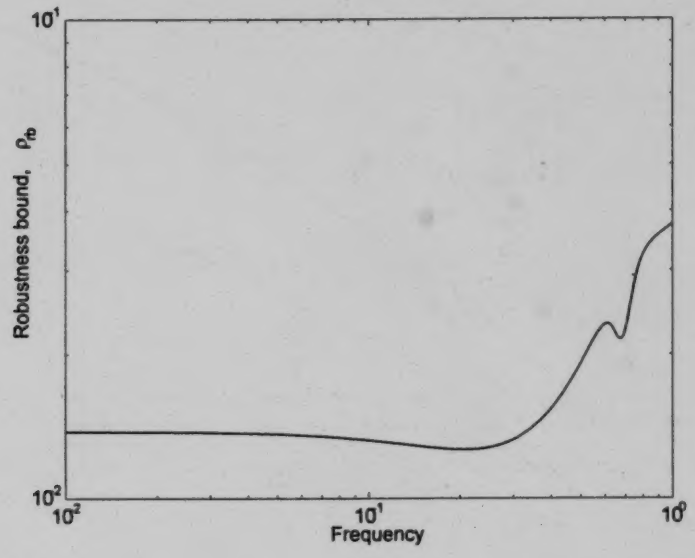


Figure 10.11: Robustness bound for the ARM-MPC

The robustness bound, ρ_{rb} for the same ARM-MPC system was determined assuming gain mismatch i.e. Δk and slow dynamics mismatch i.e. $\Delta\Phi$ as shown in Table 10.2 (a) and (b). The robustness bounds using known information about the

specific MPM in the ARM-MPC structure are compared in Figure 10.12 versus the ρ_{rb} calculated assuming the general case for MPM. It is clear from Figure 10.12 that larger bounds are obtained for known information about Δk and $\Delta\Phi$. *The reduction in the conservativeness of the robustness measure is due to the "special, dual-model structure" of the ARM-MPC.*

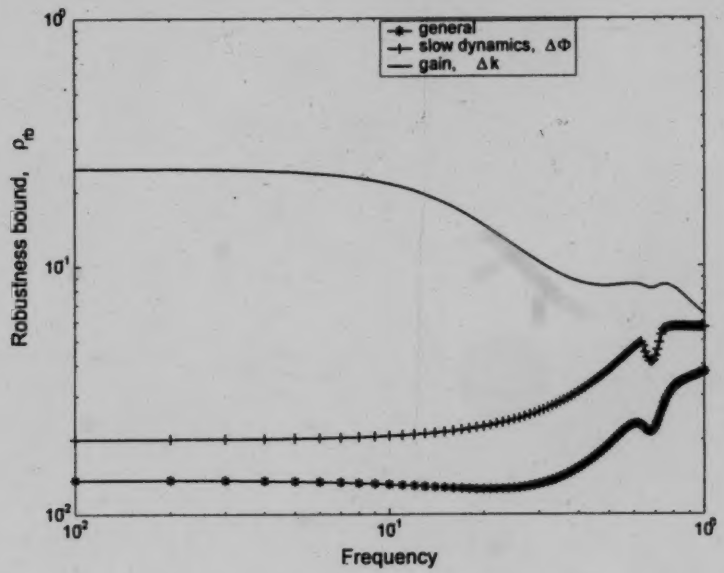


Figure 10.12: Robsutness bound for the ARM-MPC with specific MPM.

The effects of control horizon and prediction horizon on the robustness bound are shown in Figure 10.13 (a) and (b) respectively. As expected,

- ρ_{rb} decreases with an increase in M (i.e. the system is less robust) and
- ρ_{rb} increases with an increase in N_2 (i.e. the system is more robust).

However, the changes are observed mainly in the high frequency zone.

10.7 Conclusions

- The proposed closed-loop ARMarkov approach gives *multiple* performance measures including minimum variance (MV) benchmark, closed-loop settling time,

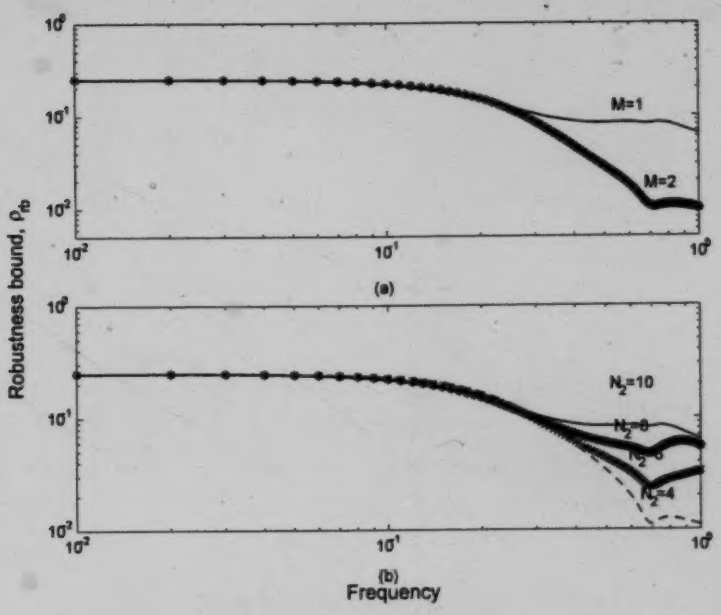


Figure 10.13: Effect of tuning parameters on the robustness bound of ARM-MPC: (a) control horizon M, (b) prediction forizon N_2.

overshoot etc. The sensitivity and complementary sensitivity functions are directly available from the full ARMarkov model and also provide closed-loop performance measures as well as an estimate of the process model.

- The user-specified LQG performance benchmark provides a direct method for evaluating the performance of different ARM-MPC's and is a key component of the robustly stable tuning procedure developed in the next chapter.

- The special structure, dual-model state space equivalent of the ARM-MPC expresses the system uncertainty as perturbations in the fast dynamics θ_{arm} or in the slow dynamics Φ_{arm} or gain/time-delay mismatch which provides a specific and/or different perspective than classical perturbation methods such as additive parametric uncertainty, $A + \Delta A$. Furthermore, this special state space structure leads to larger robustness bounds and less conservative robustness criteria than the classical small gain theorem (SGT).

Chapter 11

Stable On-line Tuning of ARM-MPC for Performance-Robustness Trade-offs

11.1 Introduction

The ultimate goal of designing a controller is to obtain good performance as well as a closed loop system that is robust to model uncertainties and disturbances. Once the controller is designed and implemented on the process, it is customary to adjust the controller parameters rather than re-design the controller to obtain the desired performance/robustness. It is important that the controller tuning or parameter adjustment can be performed on-line so that the change can be done without interfering with the ongoing process operation.

Qi (1997)discussed the dynamic tuning of his dual-model predictive control (DMPC) using what he called "a fractional control horizon parameter α". The outputs from two independent controllers designed using different control horizons were combined using the $0 \leq \alpha \leq 1$ parameter which was shown to be equivalent to changing the control horizon continuously from M to $M+1$. In this thesis, this concept is extended in an ARM-MPC context for on-line tuning. The $\alpha-$controller formulation and its properties are posed in a general form in Sections 11.2 and 11.3. Then two specific methods are developed for efficient design and implementation of the two different controllers required to obtain two individual controller outputs. The methods used

are: (1) recursive calculation of controller outputs for two different control horizons as described in Section 11.5 and (2) normalized 'zero' and 'one' steady-state weightings that define specific controllers (e.g. very aggressive and conservative controllers) as discussed in Section 11.6. Both approaches eliminate the need to do two individual controller calculations at each control interval and use simple algebraic calculations or updates of the calculations done in the previous step. The α parameter is used to combine the controller outputs on-line to produce the desired closed-loop performance /robustness. The $\alpha-$tuning is discussed in Section 11.4.

It is shown that, under the conditions given in Lemma 11.1, if each controller used individually gives robustly stable performance then the closed-loop system is robustly stable for all α and that as the value of α increases, the feedback gain changes smoothly between the limits defined by the two separate controllers. When the α parameter is used to linearly combine two controller outputs, a performance/robustness trade-off curve is constructed similar to that of Figure 10.5. Different combinations of performance/robustness can then be obtained along that curve by adjusting $0 \leq \alpha \leq 1$. A performance/robustness surface is introduced in Section 11.8 and it is shown that using a combination of ≥ 3 controllers, a *surface* with different controller output combinations can be obtained. The performance at each combination point is assessed relative to an LQG benchmark and the robustness is measured by using the matrix perturbation method as discussed in Chapter 10. Simulation results are presented for each method of controller calculation.

11.2 Continuous Control Action

Typically the aggressiveness or the conservativeness of the closed-loop system is determined by the controller output signal(s). The performance/robustness of the closed-loop system can, therefore, be controlled by adjusting the controller output(s). In this thesis it is proposed that the control outputs from two Model-based Predictive Controllers (MPC) be combined linearly using a continuous parameter $0 \leq \alpha \leq 1$. As α is varied from 0 to 1 the system properties vary continuously between the limits defined by each controller used individually e.g. from aggressive, high-performance to conservative, robust control. A linear interpolation can be done along the straight line

$A - B$ in Figure 11.1 between the two control vectors $\Delta \mathbf{u}_m^0$(point A, for controller 1 with controller gain K_{mpc}^0) and $\Delta \mathbf{u}_m^1$(point B, for controller 2 with gain K_{mpc}^1). Defining a real parameter, $\alpha \in [0, 1]$, and the control move $\Delta \mathbf{u}_m^\alpha$, leads to

$$\Delta \mathbf{u}_m^\alpha = \Delta \mathbf{u}_m^0 + \alpha \left(\Delta \mathbf{u}_m^1 - \Delta \mathbf{u}_m^0 \right) \tag{11.1}$$

Therefore, the continuously adjustable parameter, α, can be used to vary the input, $\Delta \mathbf{u}_m^\alpha$ to the process and in turn, tune controller performance/robustness.

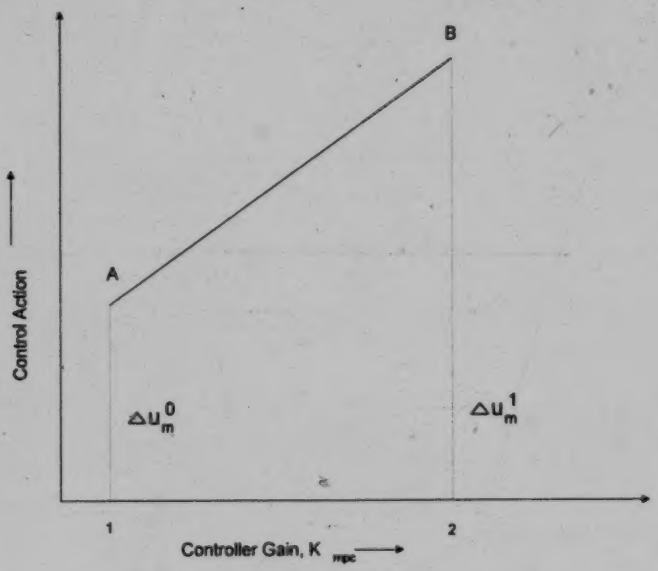


Figure 11.1: Control Actions and Interpolations

11.3 Stability of the α−controller

As shown in Chapter 8, the control move is calculated in terms of the state feedback gain K_{mpc} as

$$\Delta \mathbf{u} = -K_{mpc} X^*(k) \tag{11.2}$$

where X^* is the estimated system state vector. Typically, the larger the controller gain the more aggressive and less robust is the controller. The control action of the

$\alpha-$controller, expressed in terms of $\Delta \mathbf{u}_m^\alpha$, is bounded by the two conventional MPC controllers with control actions $\Delta \mathbf{u}^0$and $\Delta \mathbf{u}^1$. Stability analysis can be applied to this algorithm as discussed below.

11.3.1 Controller Gain Calculation

As discussed in Chapter 8, in the state space formulation, the controller gain vector K_{mpc} of the state feedback MPC controller is a function of the tuning parameters λ, M, N_2 and the weighting terms in the control objective function. Closed-loop stability of the MPC systems depends on the locations of the closed-loop poles which can be expressed in terms of the eigenvalues of the closed-loop system matrix $\Phi - \theta K_{mpc}$.

Suppose K_{mpc}^0 is the gain of controller 1 with control move $\Delta \mathbf{u}^0$ and K_{mpc}^1 is the gain of controller 2 with control move $\Delta \mathbf{u}^1$. Then the gain of the $\alpha-$controller corresponding to the control calculation in (11.1) is

$$K_{mpc}^\alpha = K_{mpc}^0 + \alpha \left(K_{mpc}^1 - K_{mpc}^0 \right) \tag{11.3}$$

If the value of α is specified, K_{mpc}^α can be calculated and the closed-loop stability can be evaluated as discussed below.

11.3.2 Stability of the $\alpha-$MPC

For closed-loop stability, all the eigenvalues of the CL system matrix $\Phi - \theta K_{mpc}$ must lie within the unit circle. The closed-loop stability of the $\alpha-$MPC with controller gain K_{mpc}^α corresponding to the controller output $\Delta \mathbf{u}^\alpha$ is given by the following Lemma.

Lemma 11.1 *The closed-loop MPC system with continuous parameter α is stable for all $\alpha \in [0,1]$, if the limiting two MPC systems are stable and there exists an induced norm $\|\cdot\|$ such that*

$$\begin{aligned} \left\| (\Phi - \theta K_{mpc}^0) \right\| &< 1 \\ \text{and } \left\| (\Phi - \theta K_{mpc}^1) \right\| &< 1 \end{aligned} \tag{11.4}$$

Proof. *Sufficiency:*

For the MPC controller with gain K_{mpc}^0, $\left\| \Phi - \theta K_{mpc}^0 \right\| < 1$

and for the MPC controller with gain K^1_{mpc}, $\left\|\Phi - \theta K^1_{mpc}\right\| < 1$

Now following the proof given by Qi (1997),

$$\begin{aligned} \left\|\Phi - \theta K^{\alpha}_{mpc}\right\| &= \left\|\alpha\left(\Phi - \theta K^1_{mpc}\right) + (1-\alpha)\left(\Phi - \theta K^0_{mpc}\right)\right\| \quad (11.5) \\ &\leq \alpha\left\|\left(\Phi - \theta K^1_{mpc}\right)\right\| + (1-\alpha)\left\|\left(\Phi - \theta K^0_{mpc}\right)\right\| \\ &< \alpha + (1-\alpha) = 1 \end{aligned}$$

$$\textit{Therefore,}\ \left\|\Phi - \theta K^{\alpha}_{mpc}\right\| < 1 \quad (11.6)$$

Since $max(eig\left(\Phi - \theta K^{\alpha}_{mpc}\right))$ *is less than any induced norm,*

$$\begin{aligned} max(eig\left(\Phi - \theta K^{\alpha}_{mpc}\right)) &\leq \left\|\Phi - \theta K^{\alpha}_{mpc}\right\| \\ &< 1 \end{aligned}$$

Therefore, all the eigenvalues of $\left(\Phi - \theta K^{\alpha}_{mpc}\right)$ *lie inside the unit circle and hence the MPC controller with gain* K^{α}_{mpc} *is stable.* ■

Consider a stable controller with gain K^1_{mpc}. The sufficient condition for this controller to be stable is

$$\begin{aligned} \left\|\left(\Phi - \theta K^1_{mpc}\right)\right\| &< 1 \quad (11.7) \\ \text{or equivalently } \left\|\Phi + \left(-\theta K^1_{mpc}\right)\right\| &< 1 \end{aligned}$$

and a sufficient, but more restrictive condition for this can be written as

$$\|\Phi\| + \left\|\theta K^1_{mpc}\right\| < 1 \quad (11.8)$$

This condition holds if

$$\|\Phi\| + \|\theta\|\left\|K^1_{mpc}\right\| < 1 \quad (11.9)$$

Therefore, under the more restrictive condition of (11.9), Lemma 11.1 can be re-stated as:

Lemma 11.2 *If* $\|\Phi\| + \|\theta\|\left\|K^1_{mpc}\right\| < 1$ *and there exists* ***any*** *controller such that* $\left\|K^0_{mpc}\right\| < \left\|K^1_{mpc}\right\|$ *for the same* Φ *and* θ*, then for the continuous parameter* $\alpha \in [0,1]$*, the controller with gain* K^{α}_{mpc} *(i.e. controller output* $\Delta \mathbf{u}^{\alpha}_m$*) is stable.*

Proof. *Using the linear combination of controller gains in (11.3),*

$$\left\|K^0_{mpc}\right\| \leq \left\|K^\alpha_{mpc}\right\| \leq \left\|K^1_{mpc}\right\| \tag{11.10}$$

Then the proof directly follows from equation (11.9). ■

Remark 11.1 *The formulations developed in this section for the α-controller are valid for both SISO and MIMO systems. This is obvious because Lemmas 11.1, 11.2 and their proofs are applicable to both SISO and MIMO systems which follows directly from the state space form.*

Remark 11.2 *Only the unconstrained solutions to the α-controller are discussed in this thesis. Hard constraint handling is beyond the scope of this thesis. However, Qi (1997)proposed an algorithm to handle hard constraints using a continuous control horizon. The general idea was "to continuously adjust the α parameter to avoid active constraints".*

11.4 Stable On-line tuning Using the α Parameter

Once the two stabilizing MPC controllers with control moves $\Delta\mathbf{u}^0$ and $\Delta\mathbf{u}^1$ are available, the closed-loop performance of these controllers can be assessed on-line against an LQG performance curve as illustrated in Figure 11.2.

Suppose controller 1 with control move $\Delta\mathbf{u}^0$ and gain K^0_{mpc} is the conservative controller and controller 2 with control move $\Delta\mathbf{u}^1$ and gain K^1_{mpc} is the aggressive controller. In this case, typically $\left\|K^1_{mpc}\right\| > \left\|K^0_{mpc}\right\|$. Then using the α parameter the control moves and in turn, the controller gains are combined linearly. The connecting line $A - B$ in Figure 11.1 will map into a performance/robustness trade-off curve $A1 - B1$ in Figure 11.2 since each point on this curve represents the performance/robustness of a particular $\alpha-$controller. The location of the points $A1$ and $B1$ relative to the user-specified LQG performance curve is determined by the nominal design of the two, individual reference controllers, e.g. by the parameters λ, N_2, M. The distances from the points $A1$ and $B1$ to the LQG performance curve are direct measures of the controller performance relative to this user-specified benchmark. The shape of the curve is a function of the process dynamics. Typically, as the controller gain increases, improved performance (lower var(y)) is obtained at the cost of

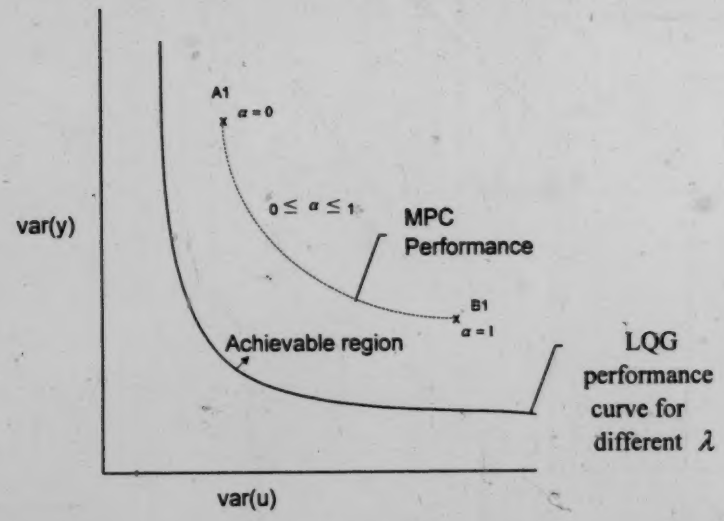


Figure 11.2: Tuning of the α controller against the LQG performance curve.

robustness (larger var(u)) i.e. a performance increase usually means a decrease in robustness. The performance/robustness of the controller can be adjusted by choosing the appropriate value of α.

11.5 Controller Design Using a Continuous Control Horizon

The control horizon, M is a very effective parameter in MPC tuning as illustrated by Figure 10.6. However, since M in classical MPC design is an integer tuning parameter, the controller gain K_{mpc} and hence the controller performance for different M changes in a step-wise fashion. When using the α parameter tuning of MPC, the two limiting controllers with gains K^0_{mpc} and K^1_{mpc} can be designed for two control horizons $M = m$ and $M = m + n$. *For this case, varying α from 0 to 1 is equivalent to varying the control horizon continuously from $M = m$ to $M = m + n$. The tuning effects are therefore smooth, bounded by known (nominal) limits, easy to interpret and apply to both SISO and MIMO systems.*

11.5.1 Recursive Calculation of Control Outputs for Two Control Horizons

To avoid two separate control calculations at each control interval, an analytical expression is developed so that the MPC output with a control horizon of $M = m+n$ can be easily calculated from the output with a horizon of $M = m$ by the addition of a simple "update term". Qi (1997)developed the recursive control move calculation with control horizons $M = m$ and $M = m + 1$. It is discussed in Chapter 8 in the ARM-MPC context and briefly revisited below.

For an MPC with a control horizon, $M = m$, denote the dynamic matrix as G_m and the present/future control move vector as $\hat{\mathbf{u}}_m^0$. Assume that an extra control term Δu_{m+1} is added to the MPC designed with a control horizon $M = m$. The new dynamic matrix G_{m+1} can be expressed in terms of the previous matrix G_m and a new coefficient vector x_{m+1} as:

$$G_{m+1} = [G_m,\ x_{m+1}] \tag{11.11}$$

where

$$x_{m+1} = [0, \cdots, 0, s_1, \cdots s_{N_2-m}]^T_{(N_2\times 1)} \tag{11.12}$$

The least-squares solution to the future control problem can, then, be written as

$$\begin{bmatrix} G_m^T G_m + \lambda I & G_m^T x_{m+1} \\ x_{m+1}^T G_m & x_{m+1}^T x_{m+1} + \lambda_{m+1} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}_m^1 \\ \Delta u_{m+1} \end{bmatrix} = \begin{bmatrix} G_m^T \\ x_{m+1}^T \end{bmatrix} E \tag{11.13}$$

The control vector, $\hat{\mathbf{u}}_m^1$ containing $m + 1$ future control moves (in terms of the $m-$vector solution, $\hat{\mathbf{u}}_m^0$) is

$$\hat{\mathbf{u}}_m^1 = \hat{\mathbf{u}}_m^0 - G_1 x_{m+1}^T \left(E - G_m \hat{\mathbf{u}}_m^0\right) \tag{11.14}$$

$$\Delta u_{m+1} = G_2 x_{m+1}^T \left(E - G_m \hat{\mathbf{u}}_m^0\right) \tag{11.15}$$

where

$$\hat{\mathbf{u}}_m^0 = \left(G_m^T G_m + \lambda I\right)^{-1} G_m^T E \tag{11.16}$$

$$G_1 = G_3 G_2 \tag{11.17}$$

$$G_2 = \left(x_{m+1}^T x_{m+1} + \lambda_{m+1} - x_{m+1}^T G_m G_3\right)^{-1} \tag{11.18}$$

$$G_3 = \left(G_m^T G_m + \lambda I\right)^{-1} G_m^T x_{m+1} \tag{11.19}$$

See Chapter 8 for proof and detailed derivation.

11.5.2 Continuous Control Horizon from $M = m$ to $M = m+n$

For predictive controllers with control horizons $M = m$ and $M = m + n$ as opposed to the special case of $M = m$ and $M = m + 1$ (line $A - B$ in Figure 11.3), a linear interpolation between the two control vectors can be obtained which corresponds to a control horizon $M = m+\alpha n$ (line $A-C$ in Figure 11.3). The control move update, in this case, includes a (n by n) matrix inversion $\left(G_2^{-1}\right)$ as discussed next.

For two controllers with control horizons $M = m$ and $M = m + n$, equations (11.11-11.19) can be re-written as

$$G_{m+n} = [G_m,\ x_{m+n}] \tag{11.20}$$

where

$$x_{m+n} = \begin{bmatrix} s_1 & 0 & \cdots & 0 & 0 \\ s_2 & s_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ s_M & s_{M-1} & \cdots & \cdots & s_1 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ s_{N_2-m} & s_{N_2-m-1} & \cdots & \cdots & s_{N_2-m-n+1} \end{bmatrix}_{(N_2 \times n)} \tag{11.21}$$

The least-squares solution to the future control problem can then be written as

$$\begin{bmatrix} G_m^T G_m + \lambda I & G_m^T x_{m+n} \\ x_{m+n}^T G_m & x_{m+n}^T x_{m+n} + \lambda_{m+1} I_{n\times n} \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{u}}_m^n \\ \Delta \mathbf{u}_{m+n} \end{bmatrix} = \begin{bmatrix} G_m^T \\ x_{m+n}^T \end{bmatrix} E \tag{11.22}$$

The control vector, $\widehat{\mathbf{u}}_m^n$ containing $m + n$ future control moves (in terms of the $m-$vector solution, $\widehat{\mathbf{u}}_m^0$) is

$$\widehat{\mathbf{u}}_m^n = \widehat{\mathbf{u}}_m^0 - G_1 x_{m+n}^T \left(E - G_m \widehat{\mathbf{u}}_m^0\right) \tag{11.23}$$

$$\Delta \mathbf{u}_{m+n} = \begin{bmatrix} \Delta u_{m+1} \\ \vdots \\ \Delta u_{m+n} \end{bmatrix} = G_2 x_{m+n}^T \left(E - G_m \widehat{\mathbf{u}}_m^0\right) \tag{11.24}$$

where

$$\widehat{\mathbf{u}}_m^0 = \left(G_m^T G_m + \lambda I\right)^{-1} G_m^T E \tag{11.25}$$

$$G_1 = G_3 G_2 \tag{11.26}$$

$$G_2 = \left(x_{m+n}^T x_{m+n} + \lambda_{m+1} I_{n\times n} - x_{m+n}^T G_m G_3\right)^{-1} \tag{11.27}$$

$$G_3 = \left(G_m^T G_m + \lambda I\right)^{-1} G_m^T x_{m+n} \tag{11.28}$$

The linear interpolation between the control moves can be written as

$$\Delta \mathbf{u}_m^{\alpha} = \Delta \mathbf{u}_m^0 + \alpha \left(\Delta \mathbf{u}_m^n - \Delta \mathbf{u}_m^0 \right) \tag{11.29}$$

$$= \Delta \mathbf{u}_m^0 - \alpha K_m \left(E - G_m \Delta \mathbf{u}_m^0 \right) \tag{11.30}$$

where $K_m = G_1 x_{m+n}^T$. It is clear that the control action, $\Delta \mathbf{u}_m^{\alpha}$, is a linear combination of the two conventional MPC controllers, *e.g.* those with control horizons $M = m$ and $M = m + n$.

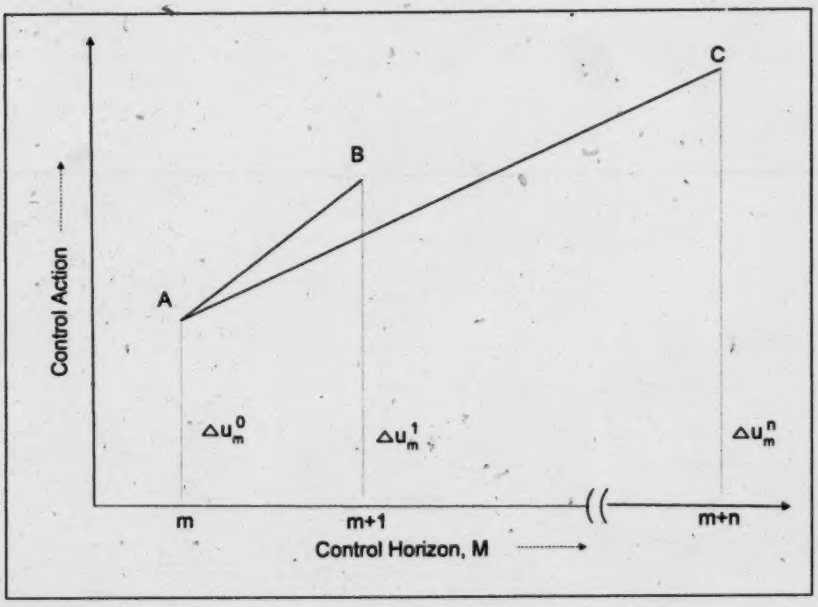


Figure 11.3: Control Actions and Interpolations for Different Control Horizons

Remark 11.3 *The control horizon concept described here is valid for both SISO and MIMO systems without any difficulty because both* $\Delta \mathbf{u}_m^1$ *and* $\Delta \mathbf{u}_m^0$ *are vectors of the same size* $(r * M \times 1)$ *where* r *is the number of inputs and* M *is the control horizon and the linear interpolation between* $\alpha = 0$ *and* $\alpha = 1$ *is valid for both the systems. This is shown in Chapter 8.*

In (11.3) K_{mpc}^0 is the controller gain with control horizon $M = m$; and K_{mpc}^1 is the controller gain with control horizon $M = m + 1$ or $M = m + n$. '*For integer changes*

in the MPC control horizon, the norm of the controller gain changes as a stair type function. However, use of the above combination of controller outputs gives smooth changes in the controller gain, i.e. like the interpolation line in Figure 11.1 and can be interpreted as tuning the MPC by changing the control horizon continuously from $M = m$ *to* $M = m + n$*'.*

As discussed in Section 8.2, all the lemmas are valid for any two controllers with gains K^0_{mpc} and K^1_{mpc}. However, the two MPC controllers with two different control horizons can be calculated recursively with less computational effort.

11.5.3 Simulation Results

The results developed in the previous sections are illustrated by the following simulation examples.

Example 11.1

Consider the third order process

$$P(z) = \frac{0.0077z^{-3} + 0.0212z^{-4} + 0.0036z^{-5}}{1 - 1.9031z^{-1} + 1.1514z^{-2} - 0.2158z^{-3}} \tag{11.31}$$

An ARM-MPC is designed as described in Chapter 8 with $\mu = 10$, prediction horizon, $N_2 = 10$ and the parametric model order, $n = 3$. The control weighting and the output weighting are 0 and 1 respectively. As the continuous horizon parameter, α, described in Section 11.2, is increased from 0 to 1 the control horizon changes from $M = 1$ to $M = 2$. The closed-loop response for this process shown in Figure 11.4 shows that (as expected) the control action becomes stronger and the output response becomes faster as α increases from 0 to 1. This is confirmed by the performance with respect to the user-specified LQG benchmark shown in Figure 11.5.

It is also anticipated that the robustness to uncertainties will decrease. This is confirmed by the robustness bounds calculated using the matrix perturbation method (described in Chapter 10) and plotted in Figure 11.6.

It is clear in this example that as the value of α is increased, the controller performance (in turn, the aggressiveness) increases whereas the robustness of the controller decreases. This illustrates how the controller can be easily tuned on-line to obtain performance/robustness trade-off.

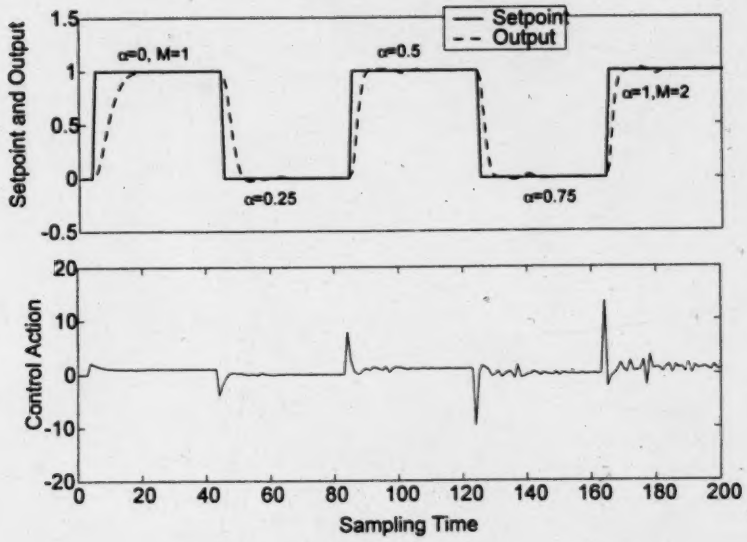


Figure 11.4: Closed-loop response of a non-minimum phase process by changing α

11.6 On-line Tuning Using Steady State Weighting

As discussed in Chapter 8 steady-state(SS) weighting gives the effect of increasing the prediction horizon without actually changing N_2 and can provide robust stability for predictive controllers. Saudagar (1995) showed that the steady-state weighting parameter γ_∞ gives better stability properties than the move suppression factor or control weighting λ. Saudagar (1995) defined a normalized γ_∞ concept where $\gamma_\infty = 0$ means no SS weighting and $\gamma_\infty = 1$ means a "heavy" SS weighting. Using appropriate design methods, $\gamma_\infty = 0$ and $\gamma_\infty = 1$ can also be considered as "aggressive, high performance" and "conservative, more robust" controllers respectively. Calculation of the SS gain using the ARMarkov model, state space ARM-MPC formulation and control calculation are discussed in Chapter 8. The steady state gain using the ARMarkov model is given in equation (8.61) as

$$s_\infty = H_{arm}\left(\Phi_1 - I\right)^{-1}\theta_1 = \frac{1}{(1+\alpha_1+\cdots+\alpha_n)}\left[\,1+\alpha_1 \quad 1 \quad 1 \quad \cdots \quad 1\,\right]\theta_1$$

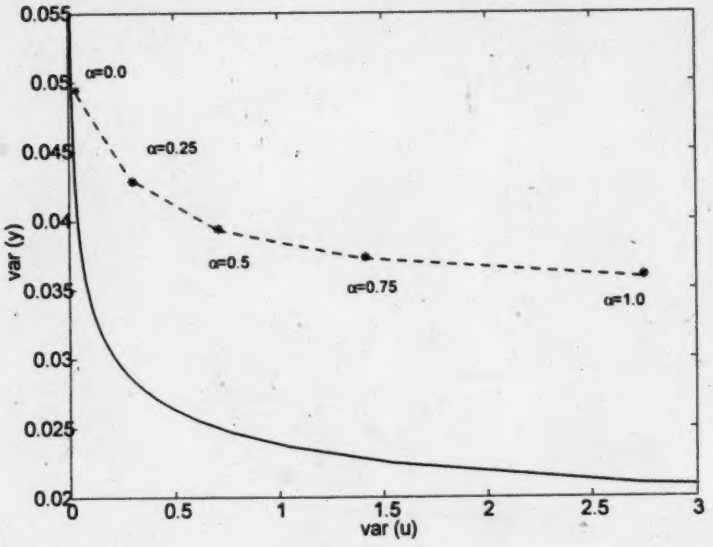


Figure 11.5: Performance of the ARM-MPC relative to the LQG benchmark for different values of α.

and the control solution using the SS weighting is

$$\begin{aligned} \hat{\mathbf{u}} &= \left[G_\infty^T Q_\infty G_\infty + \Lambda_\infty\right]^{-1} G_\infty^T Q_\infty E_\infty \qquad (11.32) \\ \text{where } E_\infty &= \left(Y_{sp(a)} - Y_\infty(k)\right) = \begin{bmatrix} E \\ e_\infty \end{bmatrix} \\ e_\infty &= y_{sp(\infty)} - \hat{y}\left(k+\infty|k\right) \end{aligned}$$

Typically, the controller gain is calculated off-line while using the controller on-line. Assume the gains for the two controllers using $\gamma_\infty = 0$ and $\gamma_\infty = 1$ are calculated off-line as

$$K_{mpc}^{\gamma_\infty=0} = \left[G^T Q G + \Lambda\right]^{-1} G^T Q \qquad (11.33)$$

$$\text{and } K_{mpc}^{\gamma_\infty=1} = \left[G_\infty^T Q_\infty G_\infty + \Lambda\right]^{-1} G_\infty^T Q_\infty \qquad (11.34)$$

where

$$\begin{aligned} G_\infty &= \begin{bmatrix} G \\ x_\infty \end{bmatrix}_{(N_2+1)\times M} \\ x_\infty &= \left[s_\infty, 0_{1\times(M-1)}\right]_{1\times M} \end{aligned}$$

$$Q_\infty = \begin{bmatrix} Q & 0 \\ 0 & \gamma_\infty \end{bmatrix}_{M\times(N_2+1)}$$

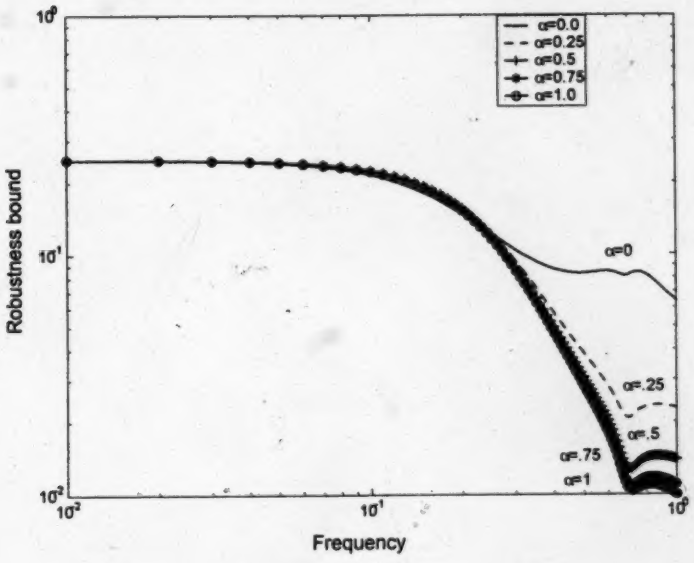


Figure 11.6: Robustness bound of ARM-MPC for different values of α

The controller with $\gamma_\infty = 1$ and gain $K_{mpc}^{\gamma_\infty=1}$ can be considered as the conservative controller with gain K_{mpc}^{0} corresponding to point $A1$ in Figure 11.2. The controller with $\gamma_\infty = 0$ and gain $K_{mpc}^{\gamma_\infty=0}$ corresponding to the aggressive controller with gain K_{mpc}^{1} in Section 11.2 and point $B1$ in Figure 11.2. Then all the theoretical developments in Section 11.3 are applicable to these two limiting controllers and the α parameter can be used on-line as described in Section 11.4 for the tuning of the controller to obtain different performance/robustness combinations. The limiting controllers in this case represent two user-specified control applications e.g. high performance with $\alpha = 0$, and conservative, more robust with $\alpha = 1$. For non-zero α, the α-controller is guaranteed stable if the conditions in Lemmas 11.1 or 11.2 are met. System performance can be tuned on-line without significant additional computation at every sampling time since there is no matrix inversion etc.

The stability of the MPC typically increases with increasing $\gamma_\infty, \gamma_\infty \in [0, 1]$, due to the detuning of the controller. As shown by Saudagar (1995), the detuning effect of γ_∞ is higher in the lower range of γ_∞ i.e. the stability increase for $\gamma_\infty = 0$ to $\gamma_\infty = 0.2$ is much higher than the stability margin increase for the γ_∞ range $0.5 - 0.9$.

The changes in the controller gain of the $\alpha-$MPC controller are linear due to the

Table 11.1: Robustness bounds of ARM-MPC using different γ_∞

steady state error weighting, γ_∞	0.0	0.2	0.5	1.0
robust. bound, ρ_{rb}	0.0134	0.0661	0.0813	0.0883

linear interpolation of the controller outputs and controller gains as shown in Figure 11.1. When using the SS weightings $\gamma_\infty = 0$ and $\gamma_\infty = 1$ in the boundary controllers, with $\alpha = 0$ to $\alpha = 1$ the boundary performance/robustness are determined by the boundary controllers, but the intermediate performance/robustness varies as α is adjusted as shown in Figure 11.8.

11.6.1 Examples

Example 11.2 *Performance of ARM-MPC's with different SS weighting,* γ_∞.

ARM-MPC's were designed for the process in (11.31) as described in Chapter 8 with $\mu = 10$, prediction horizon, $N_2 = 5$ and the parametric model order, $n = 3$. The control weighting and the output weighting were 0 and 1 respectively. At sampling time 50 a step disturbance was added. The closed-loop response is shown in Figure 11.7 for different steady state weightings. The change in the robustness bound, ρ_{rb} of the controller was determined following the procedure described in Chapter 10 and is shown in Table 11.1. As expected, with increases in the value of γ_∞, the process output response becomes slower and more conservative. This is also reflected in the robustness bound i.e. ρ_{rb} increases with the increase of γ_∞. As discussed earlier, the detuning effect is higher in the lower range of γ_∞.

Example 11.3 *: $\alpha-$tuning using boundary controllers with different SS weighting,* γ_∞.

System performance relative to the LQG benchmark was evaluated for the same process in (11.31), using $\alpha-$tuning of the ARM-MPC's with $\gamma_\infty = 0$ and $\gamma_\infty = 1$ as the two limiting controllers. The results for step tracking are shown in Figure 11.8. Closed-loop performance with respect to the LQG benchmark is plotted in Figure 11.9. The effect of α on the performance of the ARM-MPC is similar to changing γ_∞ directly (Example 11.2). However, as mentioned earlier, the effect of using intermediate α values on the closed-loop performance is different although the

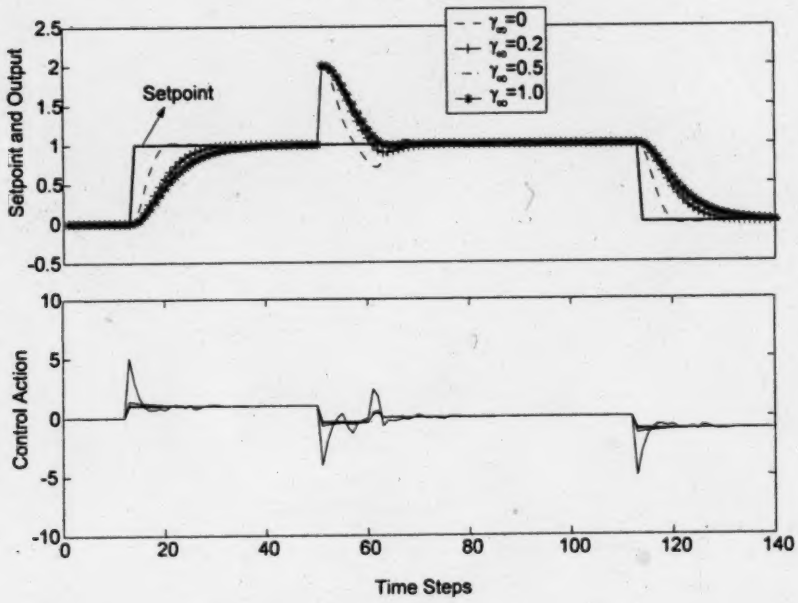


Figure 11.7: Closed-loop response for different values of steady state error weighting, γ_∞.

two limiting performances are the same as shown in Figure 11.9. As observed in Figure 11.9, the performance of the ARM-MPC using the $\alpha-$tuning is more linear than direct $\gamma_\infty-$tuning.

11.7 Trade-off Using a Combination of Steady-state Error Weighting and Control Weighting

In the preceding sections it was shown that '$\alpha-$tuning' provides (nominal) stable, on-line tuning of MPC over a user-specified range of performance/robustness that can be evaluated relative to an LQG benchmark. When the two boundary MPC's used in $\alpha-$tuning were designed using two different *control horizons* or different *SS error weightings*, the output of the second boundary MPC could be calculated by a simple extension of the first controller i.e. it was not necessary to do two independent MPC calculations at each control interval and hence $\alpha-$tuning was efficient and practical from a computational point of view.

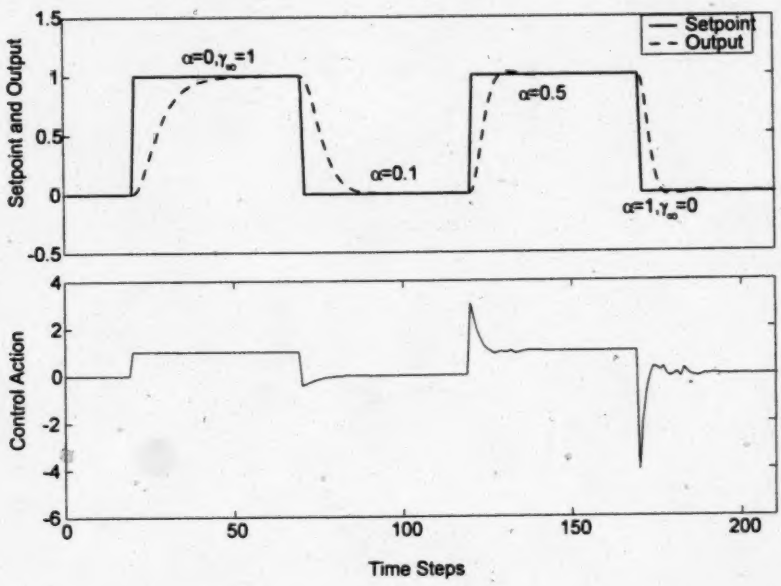


Figure 11.8: $\alpha-$tuning of the ARM-MPC using $\gamma_\infty = 0$ and $\gamma_\infty = 1$ for the two limiting controllers.

One of the most widely used parameters in the design/tuning of independent MPC's is the *control weighting*, λ and hence it is logical to ask whether it is possible to develop a computationally efficient approach for $\alpha-$tuning using boundary controllers designed using different controller weightings. Unfortunately such a controller was not found. However, the following discussion shows that it is possible to interpret $\alpha-$tuning in terms of *simultaneous* changes in SS weighting plus control weighting. More specifically, the performance obtained using the boundary controllers ($\alpha = 1$ and $\alpha = 1$) can be interpreted as MPC with two different control weightings λ_1 and λ_2. However, for $0 < \alpha < 1$ the effect is interpreted as due to a simultaneous changes in control weighting λ plus SS error weighting γ_∞, rather that a change in control weighting alone.

To develop on-line $\alpha-$tuning using *simultaneous* SS weighting and control weighting some of the results derived in Chapter 8 are used here. For example, the control move using *simultaneous* SS weighting and control weighting is discussed in Section 8.5.4.

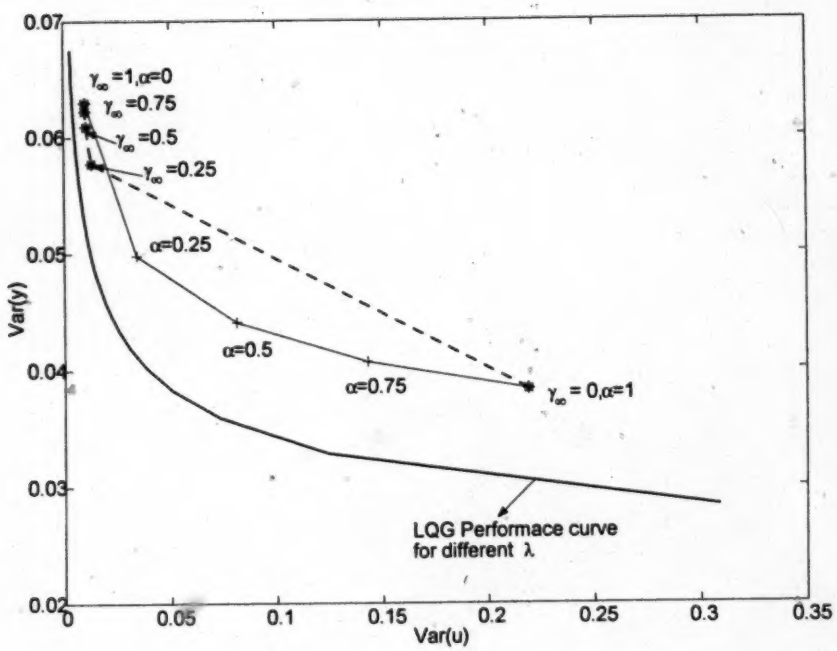


Figure 11.9: Performace comparison of the ARM-MPC for $\alpha-$tuning and γ_∞ tuning.

11.7.1 Feedback Controller Gain Update Using $\gamma_{\infty 2}$

As discussed in Chapter 8, equation (8.74) can be rewritten as

$$\Delta u(k) = C^T A_1^* \left[\begin{array}{cc} G^T Q & x_\infty^T \gamma_{\infty 2} \end{array} \right] Y_{sp(a)} - C^T A_1^* \left[\begin{array}{cc} G^T Q & x_\infty^T \gamma_{\infty 2} \end{array} \right] \Phi_{N2a} X_a$$

and the ARM-MPC feedback gain can be expressed as

$$\begin{aligned} K_{m\infty} &= a^* \left[\begin{array}{cc} G^T Q \Phi_p & x_\infty^T \gamma_{\infty 2} \end{array} \right] \\ &= \left[\begin{array}{cc} a^* G^T Q \Phi_p & a^* x_\infty^T \gamma_{\infty 2} \end{array} \right] \\ &= \left[\begin{array}{cc} K_{m\infty 1} & K_{m\infty 2} \end{array} \right] \end{aligned}$$

where

$$K_{m\infty 2} = A_1^*(1,1)\, s_\infty \gamma_{\infty 2} = a_{11}^* s_\infty \gamma_{\infty 2}$$

Keeping γ_∞ constant while varying $\gamma_{\infty 2}$ means $K_{m\infty 1}$ remains constant and the value of $K_{m\infty 2}$ varies linearly with the variation in $\gamma_{\infty 2}$. It was shown in Chapter 8

that

$$\|K_{m\infty}\|_2 \leq \|K_{m\infty 1}\|_2 + \|K_{m\infty 2}\|_2 \tag{11.35}$$

Since $\|K_{m\infty 1}\|_2$ remains constant for a constant γ_∞, the controller gain $\|K_{m\infty}\|_2$ can be updated by varying $\gamma_{\infty 2}$ and hence $\|K_{m\infty 2}\|_2$. For example, for $\gamma_{\infty 2} = 0, \|K_{m\infty}\|_2 = \|K_{m\infty 1}\|_2$ and the result is equivalent to a change in control weighting. For $\gamma_{\infty 2} = \gamma_\infty$, the result is equivalent to steady-state weighting. For values of $\gamma_{\infty 2}$ between 0 and γ_∞, an intermediate feedback gain and hence performance is obtained which can be interpreted as due to simultaneous changes in steady-state weighting and control weighting. This gives the flexibility of using either control weighting or steady state weighting or a combination of the two weighting parameters using a single tuning parameter, $\gamma_{\infty 2}$. The tuning of the controller using the $\gamma_{\infty 2}$ parameter is shown in Figure 11.10 (the model and MPC parameters are the same as in Example 11.3). The controller gain update using $\gamma_{\infty 2}$ is shown in Figure 11.11.

11.7.2 Equivalence with the $\alpha-$tuning

Let $\gamma_\infty = 0.5$. Then varying $\gamma_{\infty 2}$ from 0 to 0.5 means the controller gain is changed linearly from $K_{m\infty} = K_{m\infty 1}$ to $K_{m\infty} = K_{m\infty 1} + K_{m\infty 2}$. Therefore, considering the controller with $K_{m\infty} = K_{m\infty 1}$ as controller 1 in Section 11.2 and $K_{m\infty} = K_{m\infty 1} + K_{m\infty 2}$ as the gain of controller 2, the tuning effect of $0 \leq \gamma_{\infty 2} \leq 1$ is equivalent to the α-tuning in Section 11.2 and all the stability and performance theory are applicable. Moreover, this approach to on-line tuning updates the controller gain on-line and saves computational effort since controller re-calculation is avoided.

11.8 On-line Tuning Surface

In the previous sections, controller tuning was accomplished by linearly combining two controller outputs using the α parameter such that a performance/robustness trade-off curve was obtained. In this section, $\geq$ *three* controller outputs are combined to construct a control vector *surface* such that any two points on this surface can be used to tune the controller using the α parameter. This is illustrated in Figure 11.12 for three specific controllers. This tuning surface allows a lot more flexibility

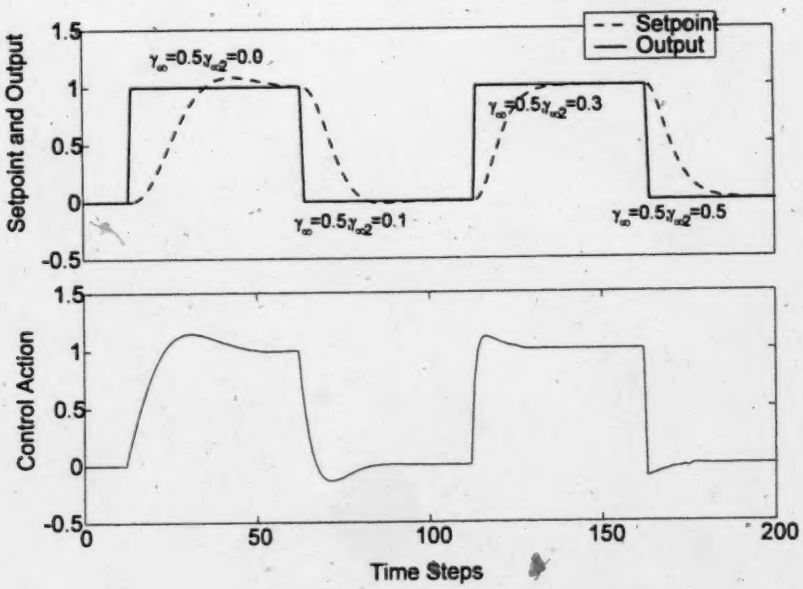


Figure 11.10: Step tracking performance of the ARM-MPC using the $\gamma_{\infty 2}$ parameter tuning.

of controller tuning than the performance/robustness trade-off curve generated by linear interpolation between two controller outputs.

Any three stabilizing controllers can be used to construct the three limiting points of the surface. For example, the three controllers can be designed with

1. control horizon $M = 1$, steady state weighting $\gamma_\infty = 0$
2. control horizon $M = 1$, steady state weighting $\gamma_\infty = 1$ (most conservative)
3. control horizon $M = 2$, steady state weighting $\gamma_\infty = 0$ (most aggressive)

Note that in this example, the output of the third controller can be obtained by updating the first controller output as discussed previously.

There are several situations in which three controller could be used to meet important operating objectives. For example, under "normal" or typical operating conditions tuning could be accomplished using two boundary MPC's and $0 \leq \alpha 1 \leq 1$ that provided a range of performance/robustness that would accommodate the process variations under normal conditions. For unusual conditions such as a major upset

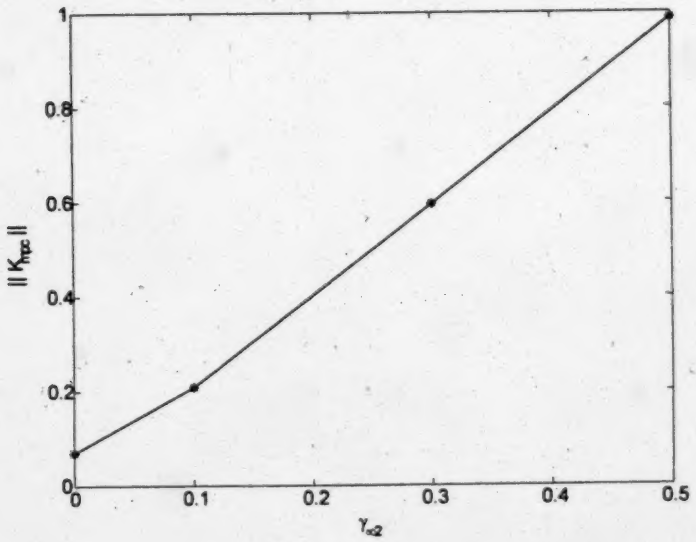


Figure 11.11: Controller gain update of the ARM-MPC using different values of $\gamma_{\infty 2}$.

or changes in operating conditions (upstream unit failure, partial equipment failure, plant utility upset etc.) the output from the "normal" controller calculated using $\alpha 1$, could be combined with the output of a third, ultra-conservative controller using $0 \leq \alpha \leq 1$ to provide "fail-safe" operation under any conditions.

11.8.1 Simulation Examples

Example 11.4

Consider the same process in (11.31). Three ARM-MPC controllers are designed using (1) $M = 1$, $\gamma_\infty = 0$, (2) $M = 1$, $\gamma_\infty = 1$, (3) $M = 2$, $\gamma_\infty = 0$. The other tuning parameters which are same for each controller are: $\mu = 10$, $N_2 = 5$, $n = 3$, control weighting $\lambda I = 0$ and output weighting $Q = I$. Controller gains for these controllers are plotted in Figure 11.13. Two different controllers (C1 and C2) are designed off-line whereas C3 is designed by updating controller C1. The step tracking performance of the three controllers are shown in Figure 11.14. Controller 2 (with $M = 1, \gamma_\infty = 1$) is the slowest (and most robust) and controller 3 (with $M = 2, \gamma_\infty = 0$) is the most aggressive controller. The $\alpha 1$ controller is tuned on-line by combining the outputs of controllers 1 and 3 by using $\alpha_1 = 0.5$. Then using $0 \leq \alpha \leq 1$, the control

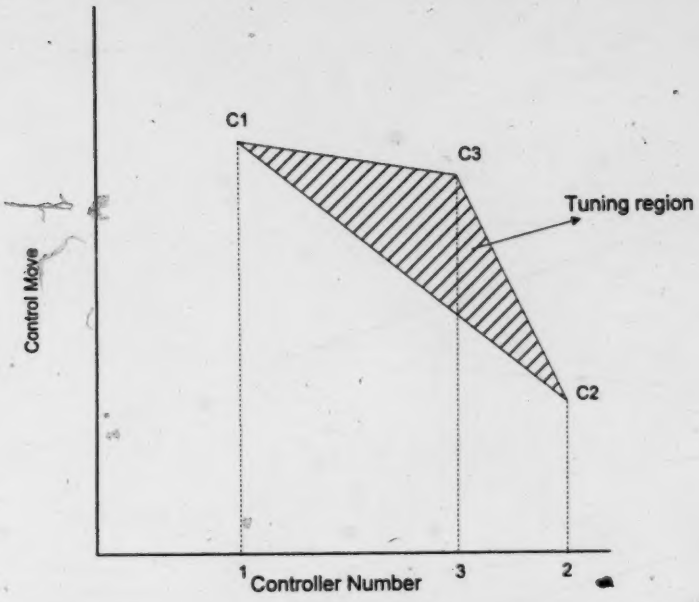


Figure 11.12: Control move surface for the $\alpha-$ tuning of ARM-MPC.

performance is tuned to obtain performance in between the performances of controller 2 and controller $\alpha 1$. The performance of this new controller for different values of α are shown in Figure 11.15. The performance of the α-controller is a combination of the performance of the three designed controllers. Comparing Figures 11.14 and 11.15, it is clear that controller performance of any degree of aggressiveness available on the tuning surface can be obtained without a significant increase in on-line computation.

11.9 Conclusions

- ARM-MPC can be effectively and efficiently tuned on-line using a single parameter $0 \leq \alpha \leq 1$ that linearly combines the outputs from two separate controllers. This is an extension of the approach used by Qi (1997)for DMPC. When both boundary controllers (i.e. for $\alpha = 0$ and $\alpha = 1$) satisfy the norm conditions in Lemmas 11.1 or 11.2, then the control obtained with $0 \leq \alpha \leq 1$ is guaranteed to be stable

- When the design of the two boundary MPC's differs only in the choice of the

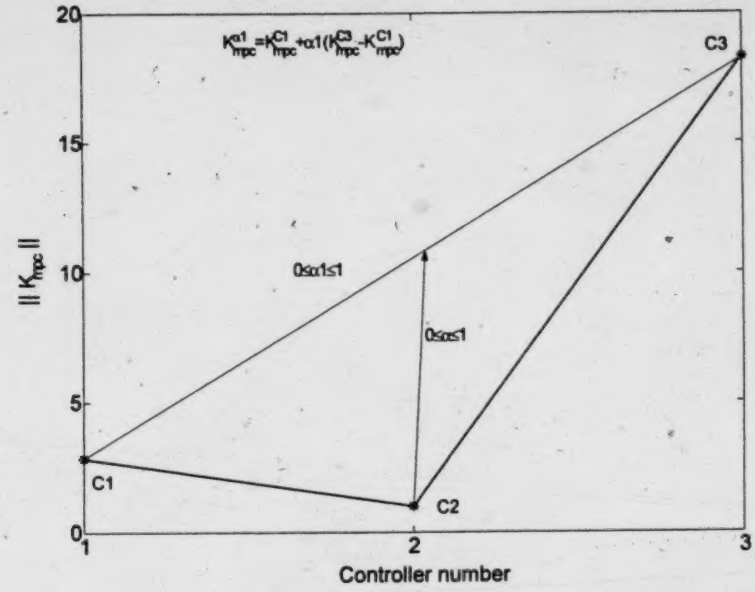


Figure 11.13: Tuning Surface constracted with 3 different controllers with different gains and controller tuning of controllers C1-C3 ($\alpha 1$ controller) and C2-$\alpha 1$ controllers.

control horizon, M or the steady state error weighting, γ_∞, then the output of the second controller can be calculated by simple, extension of the first controller calculations i.e. it is not necessary to perform two independent controller calculation at each control interval and hence $\alpha-$tuning is efficient and practical.

- The effect of the $\alpha-$tuning can also be interpreted in terms of the MPC control weighting, λ. When $\alpha = 0$, control weighting is equal to the base case with controller gain K^0_{mpc} or equivalently to using control weighting λ_0. When $\alpha = 1$, the controller gain is K^1_{mpc} or control weighting is λ_1. When the single parameter (steady state weighting) $\gamma_{\infty 2}$ is varied the effective controller gain is obtained by a simple update of the first controller gain K^0_{mpc} and the effect can be interpreted as the result of changing both the control weighting, λ and the steady state error weighting, γ_∞ simultaneously.

- The performance of the closed-loop system can be calculated for $0 \leq \alpha \leq 1$ and compared versus a user-specified LQG benchmark. For example, system

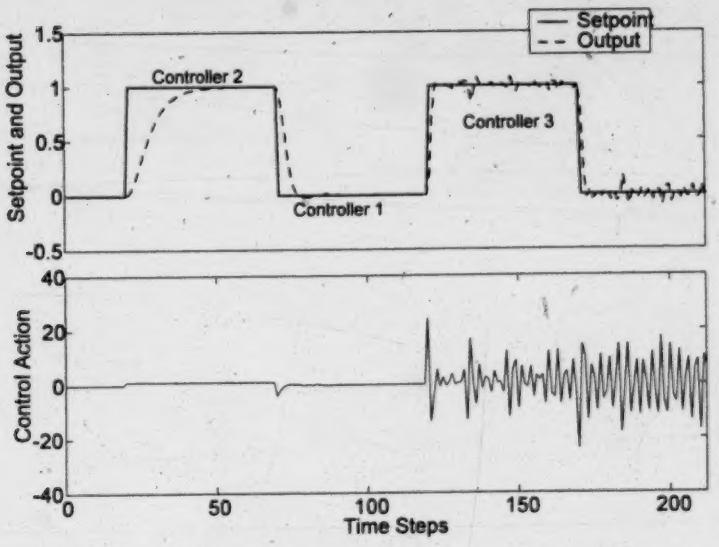


Figure 11.14: Step tracking performace of three different controllers.

performance can be varied between "aggressive" control ($\alpha = 1$ and $\gamma_\infty = 0$) and "conservative" control ($\alpha = 0$ and $\gamma_\infty =$ large) with corresponding changes in the robustness bounds. This effectively meets the overall thesis objective of developing a practical, on-line MPC tuning approach that can be used to maintain system performance relative to a user-specified LQG benchmark and/or to adjust the trade-off between robustness and performance.

- The concept of $\alpha-$tuning is extended to use three or more controllers which provide system tuning over a surface of controller outputs rather than along a single curve.

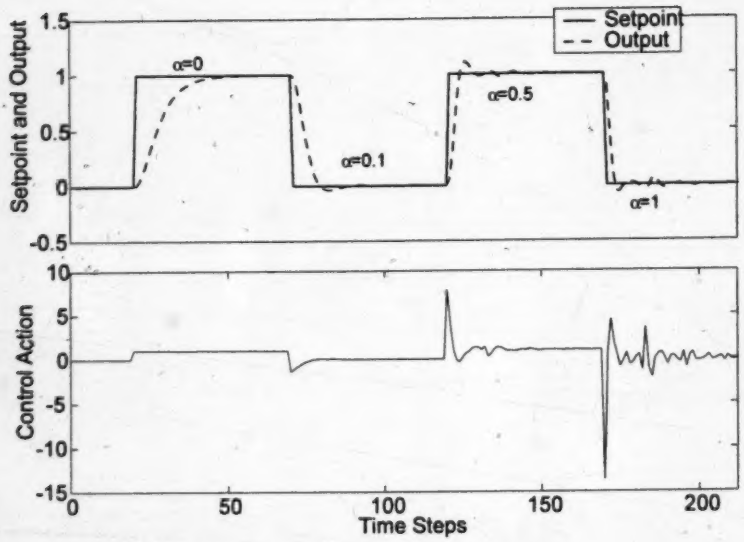


Figure 11.15: Step tracking performance of the α-controller for different values of α.

Chapter 12

Conclusions and Future Work

An ARMarkov, dual model predictive controller was developed in this thesis and is recommended for process applications such as those that occur in the chemical industry. The main areas of contribution are: (a) identification of a MIMO input-output ARMarkov model using simple least-squares methodology, (b) design and analysis of a Model-based Predictive Controller (MPC) based on the ARMarkov model and (c) on-line tuning of the controller to achieve user-specified performance/robustness trade-off(s). These contributions are summarized in the following section in the same order outlined in Table 1.1 and Figure 1.2.

12.1 Contributions

1. An input-output ARMarkov, **dual model** plus equivalent state space models were developed and used as the basis for the design, analysis, implementation and on-line tuning of a 'complete' Model-based Predictive Controller (MPC). (See Chapters 2, 3, 4, 6, 7)

    - The dual, ARMarkov model is a combination of a non-parametric and a parametric model. The non-parametric part of the model consists of a user-specified number of Markov parameters and the parametric part is a residual model that approximates the slower dynamics of the process. This model combines the advantages of FIR models (used in DMC) and ARIMAX models (used in GPC).
    - The μ Markov parameters (impulse response coefficients) define the time

delay and initial (fast) response of the system. Thus it is not necessary to specify the process time delay, a parametric model structure, or an exact model order.

- In practice, a low order ARX model is sufficient to model the parametric part of the residual or slow dynamics of the process response. A suitable order for the residual model is generated during the model identification step by using an AUDI form of least-squares.
- A disturbance model was added to the ARMarkov process model.
- The most powerful and complete set of analytical methods for MPC design and analysis is based on state space models. Therefore, the input-output ARMarkov model was converted into equivalent 'state-space' models that can be directly used for MPC design, simulation and analysis.

2. An **ARMarkov identification** algorithm was developed to directly estimate the user-specified μ Markov parameters plus the coefficients of the ARX model of the residual (slow dynamics) and the coefficients of the disturbance model. This method can be used '*a priori*' and/or 'as required' during plant operation to provide a suitable model for MPC. (See Chapters 2,5)

   - The ID method was extended to MIMO systems.
   - The 'order-recursive' AUDI identification method was re-formulated and used for selecting the order of the parametric residual model.
   - The Markov parameters estimated by the least-squares ARMarkov method were shown to be consistent and less sensitive to noise/disturbances than the impulse response coefficients in the complete response estimated by other LS methods such as FIR, ARX. The estimated ARMarkov parameters have better statistical properties such as variance/covariance, confidence bounds than the same properties of the parameters estimated by FIR, ARX.
   - The dynamic matrix, present in most MPC's, was constructed using the Markov parameters *directly* estimated using the ARMarkov method. The

statistical "consistency" of the Markov parameters leads to better accuracy and, in turn, to better control than other ID methods that convert the I/O or state-space models to step/impulse response coefficients.

- The interactor matrix for MIMO systems can be directly determined from the first $\mu-$Markov parameters identified using the ARMarkov method. Consistency of the Markov parameters leads to more accurate interactor matrix estimation.

3. A **model-based predictive controller (ARM-MPC)** was designed using classical techniques but is more flexible and efficient than many of the other MPC's described in the literature. (See Chapters 6,8,9)

   - The ARM-MPC is flexible enough to include DMC and GPC as limiting cases.
   - It needs fewer Markov parameters than conventional DMC and, unlike DMC, includes a disturbance model that allows independent on-line regulatory tuning.
   - It is less aggressive than conventional GPC and does not require solution and/or updating of Diophantine equations at every sampling instant.
   - The consistency of the Markov parameters leads to a more accurate dynamic matrix and hence better control.
   - The special structure of the state-space ARM-MPC leads to higher robustness bounds and less aggressive analysis criteria than the Small Gain Theorem (SGT).
   - The state space ARM-MPC, which is derived from and mathematically equivalent to the input-output form, can be directly used to calculate the LQG performance benchmarks.
   - A classical state space model (A,B,C,D matrices) was derived using the consistent Markov parameters obtained using ARMarkov identification and an Eigensystem Realization Algorithm (ERA). This state space model can be used for all classical design, analysis and simulation procedures.

- The disturbance model added to the ARMarkov model enhances regulatory control and tuning. It can be identified using the ARMarkov method or it can be treated as a user-specified filter.

4. Significant design and regulatory tuning flexibility was added to ARM-MPC by separating the disturbance prediction in the **feedback** path from the plant input-output model. (See Chapter 7)

    - The first of the two parts of the separated disturbance predictor (SDP) equation consists of terms that depend on the plant model. The second part depends only on the residual (error in the current output prediction) and the noise model and is completely independent of the plant model. Thus the SDP makes it easy to provide more flexible tuning such as independent 'servo' and 'regulatory' tuning of the controller through the use of a 'disturbance horizon'.

5. The proposed ARMarkov model-based **performance assessment** approach simultaneously produces multiple metrics for closed loop performance monitoring/assessment. (See Chapter 10)

    - Estimation of a reliable and accurate interactor matrix is a pre-requisite for performance assessment methods such as the conventional Minimum Variance (MV) benchmark. It can be estimated from the consistent Markov parameters directly identified by the ARMarkov method from routine closed-loop (CL) operating data.
    - Sensitivity/complementary-sensitivity functions plus time-domain metrics such as time constants are available from the closed loop ARMarkov identification. The sensitivity/complementary sensitivity function can be used for on-line performance assessment and tuning of the controller.
    - An LQG performance curve was constructed and is recommended as a relevant and practical benchmark for MPC performance assessment over the user-defined domain of operation.

6. On-line controller **tuning** was implemented using a single parameter $0 \leq \alpha \leq 1$ that permits smooth, robustly-stable tuning over a user-defined range of control and provides a *practical* basis for implementing higher level supervisory optimization functions. (See Chapter 11)

   - This tuning approach is applicable to any two controllers, including MPC(s)
   - It linearly combines the outputs of two robustly-stabilizing, user-defined controllers using a single parameter, $\alpha \in [0, 1]$
     (a) $\alpha = 0$ (MPC #0) e.g. conservative (e.g. $M = m$, $\gamma_\infty = 1$)
     (b) $\alpha = 1$ (MPC #1) e.g. aggressive (e.g. $M = m + n$, $\gamma_\infty = 0$)
     (c) $0 < \alpha < 1$ provides smooth transition from conservative to aggressive performance
   - Changing the tuning parameter α from 0 to 1 leads to smooth changes in the state feedback gain, $K^0_{mpc} \rightarrow K^1_{mpc}$.
   - Two independent controller calculations are not required at every sampling instant if the two MPC's are designed with
     (a) $M = m \rightarrow M = m + n$, or
     (b) steady state error weighting $\gamma_{\infty 2} = 0 \rightarrow \gamma_{\infty 2} = 1$
     The second controller is obtained by simple updating of the first controller and thereby saves computational effort.

7. The system is robustly stable for all $0 \leq \alpha \leq 1$ if the two limiting MPC controllers (i.e. those used when $\alpha = 0$ and $\alpha = 1$) meet the conditions of Lemma 11.1 or 11.2, i.e. the norms of the closed-loop system matrices are less than one. This is a significant characteristic because most MPC's provide only an "optimal" solution and do not guarantee stability and/or robustness even for the specified design case. Similarly, most MPC's do not include a stability guarantee when design (tuning) parameters are changed. (See Chapter 11).

   - Performance and robustness over the user-specified region of operation ($0 \leq \alpha \leq 1$) can be directly compared to and evaluated versus benchmarks such as LQG.

- The single parameter $\alpha-$tuning approach was extended from linear interpolation (i.e. along a line) to tuning over a surface (for ≥ 3 MPC's). Off-line calculation of more than one controller output may be needed (depending on the user-specified limiting controllers) for this case. The $\alpha-$controller can be tuned to obtain the desired performance/robustness combination at every point on the surface.

12.2 Overall Conclusion and Practical Significance

From an overall point of view, this thesis provides a complete and integrated design and analysis procedure for a dual-model based predictive controller (ARM-MPC) that is easy to understand, flexible to implement on industrial processes, and provides the desired performance/robustness trade-offs through (single parameter) on-line tuning.

The proposed approach for design and tuning is particularly significant from a practical point of view because it provides an effective, intuitive method for use by plant operating personnel and/or higher level supervisory/optimization algorithms. (Most MPC;s have "too many" tuning parameters in the sense that their non-linear, interactive effects make tuning very different when attempted over a wide range of operating conditions and/or applications)

The practical advantages of the ARM-MPC arise because: the tuning interval is closed and normalized ($0 \leq \alpha \leq 1$); the limiting performance/robustness is known (e.g. a conservative $C1$ combined with an aggressive $C2$); tuning is smooth (interpreted as changes in feedback gain) over the full tuning range; the performance and robustness bounds can be analyzed a *priori* and expressed or monitored on-line in terms of performance metrics or benchmarks such as LQG and MV; and a "fail-safe" fallback control procedure can be included to handle major, atypical disturbances (via multidimensional $\alpha-$tuning).

In most process control applications *regulatory* control is more important than *setpoint tracking*. Although it may seem relatively unimportant from an MPC-structure point of view the separation of the disturbance feedback and prediction path in ARM-MPC (Figure 7.1) is significant because it allows *independent* design, implementation and tuning of the *regulatory* control versus setpoint tracking func-

tions. The disturbance model/filter can be estimated or user-specified independent of the process model; practical tuning parameters such as a disturbance prediction horizon (analogous to the output prediction horizon) can be added, and concepts such as "$\alpha-$tuning" can be used to blend multiple disturbance estimation/prediction strategies.

Significant work is still required in the MPC area, particularly to handle very large systems, constraints, non-linear behavior, higher level supervisory/optimization strategies and practical, robustly-stable, on-line tuning. However, as demonstrated in the preceding chapters, the ARM-MPC offers several structural, theoretical and practical advantages relative to most MPC's in the literature.

12.3 Recommendations for Future Work

1. The statistical analysis of estimated parameters discussed in this thesis does not include disturbance models in the ARMarkov model structure. It would be interesting to analyze the statistical properties of the estimated parameters under disturbance model estimation.

2. "Control relevant identification" has drawn a lot of attention in the MPC design and application areas. It has been shown in this thesis that the ARMarkov identification uses simple linear regression to generate "consistent" parameter estimates which, in the limit, converge to the true values. In MPC, accurate modeling leads to controllers that optimize the user specified performance index. It would be interesting to compare the approach used in this thesis versus control relevant identification from the point of view of controller performance.

3. In recent years, closed-loop performance assessment/monitoring has become an active research area. There are several very useful and important theorems and tools for performance assessment, but not much has been done in the *robust* performance analysis/assessment area. Future extension of the ARM-MPC design and analysis approach developed in this thesis could include *robust* performance criteria. This could include on-line estimation of model plant mismatch (MPM); ARM-MPC design to achieve *robust* performance; on-line performance

assessment using a robust performance criterion; and tuning of the controller to achieve robust performance. The LQG performance criterion was used in this thesis as a performance benchmark for ARM-MPC. In future work, a *robust* LQG performance criterion could be used for MPC performance assessment.

4. Joint identification and control is another promising area of research. Either the model is updated/re-identified or the controller is tuned/re-designed based on the achieved closed-loop performance/robustness. It is interesting to note that even with significant MPM, higher performance may be achieved with the resulting controller. Future extensions of this thesis could include joint identification and control in the context of ARMarkov ID and ARM-MPC design/analysis.

Bibliography

Akers, J.C. and D.S. Bernstein (1997). ARMarkov Least-Squares Identification. In: *Proceedings of the American Control Conference.* Albuquerque, New Mexico, USA. pp. 186–190.

Akers, J.C. and D.S. Bernstein (1999). Time-domain Identification Using AR-Markov/Toeplitz Models with Quasi-Newton update. In: *Proceedings of the American Control Conference.* San Diego, California, USA. pp. 738–743.

Astrom, K.J. (1970). *Introduction to Stochastic Control Theory.* Academic Press. NJ.

Astrom, K.J. and B. Wittenmark (1984). *Computer Control Systems: Theory and Design.* Prentice-Hall, Englewood Cliffs. NJ.

Banerjee, P. (1996). Robustness Issues in Long-Range Predictive Control. PhD thesis. Department of Chemical and Materials Engineering, University of Alberta. Edmonton, Canada.

Banerjee, P. and S.L. Shah (1996). Simultaneous Estimation of Parameters for Different Orders of Discrete Orthonormal Function Models. *IEEE Transactions on Signal Processing* **44**(8), 2042–2054.

Bierman, G.J. (1977). *Factorization Methods for Discrete Sequential Estimation.* Academic Press. New York, USA.

Bitmead, R.R., M. Gevers and V. Wertz (1990). *Adaptive Optimal Control: The Thinking Man's GPC.* Prentice-Hall, Englewood Cliffs. NJ.

Camacho, E.F. and C. Bordons (1999). *Model Predictive Control.* Springer-Verlag London Ltd.

Campo, P.J. and M. Morari (1986). Infinity Norm Formulation of Model Predictive Control Problems. In: *Proceedings of the American Control Conference.* pp. 339–343.

Chen, T. and B¿ Francis (1995). *Optimal Sampled-Data Control Systems.* Springer-Verlag London Ltd.

Clarke, D.W. (1994). *Advances in Model-based Predictive Control.* Oxford University Press.

Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987*a*). Generalized Predictive Control - Part I. *Automatica* **23**(2), 137–148.

Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987*b*). Generalized Predictive Control - Part I & II. *Automatica* **23**(2), 137–160.

Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987*c*). Generalized Predictive Control - Part II. *Automatica* **23**(2), 149–160.

Cutler, C.R. and B.L. Ramaker (1980). Dynamic Matrix Control - A Computer Control Algorithm. In: *Proceedings of the Joint Automatic Control Conference.* number WP5-B. San Francisco, CA, USA.

Dahlquist, G and A. Bjorck (1974). *Numerical Methods.* Translated by N. Anderson, Prentice Hall. New Jersey.

De Keyser, M.C. (1991). Basic Principles of Model Based Predictive Control. In: *Proceedings of the First European Control Conference.* pp. 1753–1758.

Desborough, L. and T.J. Harris (1993). Performance Assessment Measures for Univariate Feedforward/Feedback Control. *Canadian Journal of Chemical Engineering* **71**, 605–616.

DeVries, W.R. and S.M. Wu (1978). Evaluation of Process Control Effectiveness and Diagnosis of Variation in Paper Basis Weight via Multivariate Time Series Analysis. *IEEE Transactions on Automatic Control* **23**(4), 1186–1197.

Dickman, A. (1987). On the Robustness of Multivariable Linear Feedback Systems in State-Space Representation. *IEEE Transactions on Automatic Control* **37**(5), 407–409.

Doyle, J.C. and G. Stein (1979). Robustness with observers. *IEEE Transactions on Automatic Control.*

Doyle, J.C., B.A. Francis and A.R. Tannenbaum (1992). *Feedback Cpntrol Theory.* Maxwell Publishing Company.

Eaton, J.W. and J.B. Rawlings (1992). Model Predictive Control of Chemical Processes. *Chemical Engineering Science* **47**(4), 705–720–382.

Froisy, J.B. (1994). Model Predictive Control:past, present and future. *ISA Transactions* **33**, 235–243.

Gambier, A. and H. Unbehauen (1993). A State-Space Generalized Model-based Predictive Control for Linear Multivariable Syatems and its Interrelation with the Receding Horizon LQG-Control. In: *Proceedings of the 32nd Conference on Decision and Control.* San Antonio, Texas, USA. pp. 817–822.

Garcia, C.E. and M. Morari (1982). Internal Model Control-Part I: a Unifying Review and Some New Results. *Industrial Engineering Chemical Process Design and Development* **21**, 308–323.

Garcia, C.E., D.M. Prett and M. Morari (1989). Model Predictive Control: Theory and Practice - a survey. *Automatica* **25**, 335–348.

Gauss, K.F. (1809). *Theory Motus Corporum Colestium, English Translation:Theory of the Motion of the Heavenly Bodies(1963).* Dover, New York. New York.

Genceli, H. and M. Nikolaou (1993). Robust Stability Analysis of Constrained l1-norm Model Predictive Control. *AICHE Journal* **39**, 1954–1965.

Goodwin, G.C. and K.S. Sin (1984). *Adaptive Filtering, Prediction and Control.* Prentice-Hall, Englewood Cliffs. NJ.

Harris, T.J. (1989). Assessment of Control Loop Performance. *Canadian Journal of Chemical Engineering* **67**, 856–861.

Harris, T.J., C.T. Seppala and L. Desborough (1999). A Review of Performance Monitoring and Assessment Techniques for Univariate and Multivariate Control Systems. *Journal of Process Control* pp. 1–17.

Harris, T.J., F. Boudreau and J.F. MacGregor (1996). Performance Assessment of Multivariable Feedback Controllers. *Automatica* **32**(11), 1505–1518.

Ho, B.L. and R.E. Kalman (1965). Effective Construction of Linear State Variable Models from Input/Output Data. In: *3rd Annual Allerton Conference on Circuit and System Theory*. pp. 449–459.

Horing, S. (1962). On the Optimum Design of Predictive Control Systems. In: *WESCON*. Los Angeles, CA USA.

Hrissagis, K., O.D. Crisalle and M. Sznaier (1995). Robust Design of Unconstrained Predictive Controllers. In: *Proceedings of the American Control Conference*.

Huang, B. (1997). Multivariate Statistical Methods for Control Loop Performance Assessment. PhD thesis. Department of Chemical and Materials Engineering, University of Alberta. Edmonton, Canada.

Huang, B. and S.L. Shah (1999). *Performance Assessment of Control Loops: Theory and Applications*. Springer-Verlag.

Huang, B., S.L. Shah and E.J. Kwok (1997*a*). Good, Bad or Optimal? Performance Assessment of Multivariable Process. *Automatica* **33**(6), 1175–1183.

Huang, B., S.L. Shah and Fujii H. (1997*b*). Unitary Interactor Matrix and its Estimation using Closed-loop Data. *Journal of Process Control* **7**(3), 195–207.

Huang, B., S.L. Shah and R. Miller (2000). Feedforward Plus Feedback Controller Performance Assessment of MIMO Systems. *IEEE Transactions on Control Systems Technology* **8**(3), 580–587.

Hyland, D.C. (1991). Neural Network Architecture for On-Line System Identification and Recursively Optimized Control. In: *Proc. IEEE Conf. Dec. Contr.*. Brighton, UK. pp. 2552–2557.

Johnson, R.A. and D.W. Wichern (1988). *Applied Multivariate Statistical Analysis*. Prentice-Hall, Englewood Cliffs. NJ.

Juang, J. and R. Papa (1985). An Eigensystem Realization Algorithm for Modal Parameter Identification and Model Reduction. *Journal of Guidance, Control and Dynamics* **8**(5), 620–627.

Juang, J. and R. Papa (1986). Effects of noise on Modal Parameter Identified by the Eigensystem Realization Algorithm. *Journal of Guidance, Control and Dynamics* **9**(3), 294–303.

Juang, J.N. (1994). *Applied System Identification*. Prentice-Hall, Englewood Cliffs. NJ.

Kailath, T., Ed.) (1980). *Linear Systems*. Prentice Hall Information and System Sciences Series. Prentice Hall Inc.. Englewood Cliffs, New Jersey.

Kamrunnahar, M., B. Huang and D.G. Fisher (1998). Model Predictive Control using ARMarkov/LS Identification and a Fractional Control Horizon. presented at the Annual AICHE Meeting, Miami Beach, Florida, USA.

Kamrunnahar, M., B. Huang and D.G. Fisher (2000). Estimation of Markov Parameters and Time-Delay/Interactor Matrix. *Chemical Engineering Science* **55**(17), 3353–3363.

Kishi, F.H. (1964). *Online Computer Control Techniques and their Application to Reentry Aerospace Vehicle Control.* Advances in Control Systems, Theory and Applications, Pt. 1. Academic Press. New York, USA.

Ko, B.S. and T.F. Edgar (2000). Performance Assessment of Multivariable Feedback Control Systems. In: *Proceedings of the American Control Conference.* Chicago, IL, USA. pp. 4373–4377.

Kothare, M.V., V. Balakrishnan and M. Morari (1996). Robust constrained model predictive control using linear matrix inequalities. *Automatica* **32**(10), 1361–1379.

Kozub, D.J. (1996). Monitoring and Diagnosis of Chemical Processes With Automated Process Control. *CPC V Meeting, Lake Tahoe.*

K.R., Muske, E.S. Meadows and J.B. Rawlings (1994). Stability of constrained receding horizon control with state estimation. In: *Proceedings of the American Control Conference.* Vol. 3. pp. 2837–2841.

Kwok, K.Y. (1992). Long-Range Adaptive Predictive Control: Extensions and Applications. PhD thesis. Department of Chemical and Materials Engineering, University of Alberta. Edmonton, Canada.

Kwok, K.Y. and S.L. Shah (1994). Long-Range Predictive Control with a Terminal Mathching Condition. *Chem. Eng. Sci.* **49**(9), 1287–1300.

Lee, J.H. (1996). Recent Advances in Model Predictive Control and Other Related Issues. In: *Chemical Process Control - CPC V.* Lake Tahoe, CA, USA. pp. 201–216.

Lee, J.H., M. Morari and C.E. Garcia (1994). State-space Interpretation of Model Predictive Control. *Automatica* **4**, 707–717.

Lee, J.H., M.S. Gelormino and M. Morari (1992). Model Predictive Control of Multirate Sampled-data Systems: a State-space Approach. *Int. J. Control* **55**(1), 153–191.

Li, S., K.Y. Lim and D.G. Fisher (1989). A State Space Formulation for Model Predictive Control. *AICHE Journal* **35**(2), 241–249.

Ljung, L. (1987). *System Identification: Theory for the Users.* Prentice Hall, Englewood Cliffs. New Jersey.

Ljung, L. and T. Soderstrom (1983). *Theory and Practice of Recursive Identification.* The MIT Press.

Mayne, D.Q., J.B. Rawlings, C.V. Rao and P.O.M. Scokaert (2000). Constrained model predictive controllers: stability and optimality. *Automatica* **36**, 789–814.

Meadows, E.S. and J.B. Rawlings (1993). Receding horizon control with an infinite horizon. In: *Proceedings of American Control Conference.* pp. 2926–2930.

Meadows, E.S., Henson M.A., Eaton-J.W. and J.B. Rawlings (1995*a*). Receding horizon control and discontinuous state feedback stabilization. *International Journal of Control* **62**(5), 1217–1229.

Meadows, E.S., Muske K.R. and J.B. Rawlings (1995*b*). Implementable model predictive control in the state space. In: *Proceedings of the American Control Conference.* Vol. 5. pp. 3699–3703.

Mohtadi, C. (1988). On the Role of Prefiltering in Parameter Estimation and Control. In: *Proceedings of a Workshop on Adaptive Control Strategies fo Industrial Use.* Kanannaskis, Cananda. pp. 103–144.

Morari, M. and E. Zafiriou (1989). *Robust Process Control.* Prentice Hall, Englewood Cliffs. NJ.

Morari, M. and J.H. Lee (1991). Model Predictive Control: The Good, the Bad and the Ugly. In: *Model-Based Process Monitoring and Control - Part I, Chemical Process Control - CPC IV.* Padre Island, Texas, USA. pp. 419–444.

Morari, M. and J.H. Lee (1999). Model Predictive Control:past, present and future. *Computers and Chemical Engineering* **23**, 667–682.

Muske, K.R. and J.B Rawlings (1993). Model Predictive Control with Linear Models. *AICHE Journal* **39**, 262–287.

Navratil, J.P., K.Y. Lim and D.G. Fisher (1988*a*). Disturbance Feedback in Model Predictive Control Systems. In: *Proceedings of IFAC Int. Workshop on Model Based Process Control.* Atlanta, GA, USA. pp. 63–68.

Navratil, J.P., K.Y. Lim and D.G. Fisher (1988*b*). Feedback Control Options in Model Predictive Control Systems. In: *Proceedings of IFAC Int. Workshop on Model Based Process Control.*

Nicolao, G.D., L. Magni and R. Scattolini (1996). Robust Predictive Control of Systems with Uncertain Impulse Response. *Automatica* **32**(10), 1475–1479.

Niu, S. (1994). Augmented UD Identification for Process Control. PhD thesis. Department of Chemical and Materials Engineering, University of Alberta. Edmonton, Canada.

Niu, S., D. Xiao and D.G. Fisher (1990). A Recursive Algorithm for Simultaneous Identification of Model Order and Parameters. *IEEE Transactions on ASSP* **38**(5), 884–889.

Niu, S., D.G. Fisher and D. Xiao (1992). An Augmented UD Identification Algorithm. *International Journal of Control* **56**(1), 193–211.

Ogunnaike, B.A. and W.H. Ray (1994). *Process Dynamics, Modeling and Control.* Oxford University Press. New York.

Patwardhan, R. (1999). Studies in Synthesis and Analysis of Model Predictive Controllers. PhD thesis. Department of Chemical and Materials Engineering, University of Alberta. Edmonton, Canada.

Peng, Y. and M. Kinnaert (1992). Explicit Solution to the Singular LQ Regulation Problem. *IEEE Trans. on AC* **37**, 633–636.

Qi, K. and D.G. Fisher (1993). Model Predictive Control for Open-loop Unstable Process. In: *Proceedings of American Control Conference*. San Fransisco, California, USA. pp. 796–800.

Qi, K. and D.G. Fisher (1994). Robust Stability Analysis of Model Predictive Control. In: *Proceedings of American Control Conference*. Maryland, USA. pp. 3258–3262.

Qi, K.Z. (1997). Dual - Model Predictive Control. PhD thesis. Department of Chemical and Materials Engineering, University of Alberta. Edmonton, Canada.

Qi, K.Z., M. Kamrunnahar and D.G. Fisher (2001). Stable On-line Tuning of Model Predictive Controllers. submitted.

Qin, S.J. and T.A. Badgwell (1996). An Overview of Industrial Model Predictive Control Technology. In: *Chemical Process Control - CPC V*. Lake Tahoe, CA, USA. pp. 232–256.

Qu, Z. and J.F. Dorsey (1990). Stability Robustness of Discrete Systems with Perturbatios in State Equation. In: *Proceedings of the American Control Conference*. pp. 3054–3057.

Rawlings, J.B. and K.R. Muske (1993). Stability of Constrained Receding Horizon Control. *IEEE Transactions on Automatic Control* **38**(10), 1512–1516.

Rawlings, J.B., E.S. Meadows and K.R. Muske (1994). Nonlinear Model Predictive Control: A Tutorial and Survey. In: *ADCHEM Proceedings*. Kyoto, Japan.

Richalet, J.A., J.L. Testud and J. Papan (1978). Model Predictive Heuristic Control: Application to Industrial Processes. *Automatica* **14**, 413–428.

Ricker, N.L. (1990). Model Predictive Control with State Estimation. *Ind. Eng. Chem. Res.* **29**, 374–382.

Ricker, N.L. (1991). Model Predictive Control: State of the Art. In: *in Model-Based Process Monitoring and Control - Part I, Chemical Process Control - CPC IV*. Padre Island, Texas, USA. pp. 271–296.

Rogozinski, M.W., A.P. Paplinski and M.J. Gibbard (1987). An Algorithm for Calculation of a Nilpotent Interactor Matrix for Multivariable Systems. *IEEE Transactions on Automatic Control* **32**(3), 234–237.

Rouhani, R. and R.K. Mehra (1982). Model Algorithm Control. *Automatica* **18**, 401–41.

Ruscio, D.D. (1997*a*). Model based predictive control: An extended state space approach. In: *Proceedings of the 36th Conference on Decision and Control*. San Diego, California, USA.

Ruscio, D.D. (1997*b*). Model Predictive Control and Indetification: A Linear State Space Model Approach. In: *Proceedings of the 36th Conference on Decision and Control*. San Diego, California, USA.

Sane, H.S., R. Venugopal and D.S. Bernstein (1999*a*). Disturbance Rejection Using Self-Tuing ARMarkov Adaptive Control with Simultaneous Identification. In: *Proceedings of the American Control Conference*. San Diego, California, USA. pp. 2040–2044.

Sane, H.S., R. Venugopal and D.S. Bernstein (1999*b*). Robustness of ARMarkov Adaptive Control Disturbance Rejection Algorithm. In: *Proceedings of the American Control Conference*. San Diego, California, USA. pp. 2035–2039.

Saudagar, M. (1995). Unified Model Predictive Control. PhD thesis. Department of Chemical and Materials Engineering, University of Alberta. Edmonton, Canada.

Scokaert, P.O.M., J.B. Rawlings and E.S. Meadows (1997). Discrete-time stability with perturbations: application to model predictive control. *Automatica* **33**(3), 463–470.

Shah, S.L., C. Mohtadi and D.W. Clarke (1987). Multivariable Adaptive Control without a prior Knowledge of the Delay Matrix. *Systems and Control Letters* pp. 295–306.

Skogestad, S. and I. Postlethwaite (1996). *Multivariable Feedback Control*. John Wiley and Sons.

Soderstrom, T. and P. Stoica (1989). *System Identification*. Prentice Hall International Ltd. UK.

Soeterboek, A.R.M. (1992). *Predictive Control-a Unified Approach*. Prentice Hall International Series in Systems and Control Engineering.

Sripada, N.R. and D.G. Fisher (1985). Multivariable Optimal Constrained Control Algorithm (mocca), Part 1, Formulation and Application. In: *Proceedings of International Conference on Modeling and Control*. Hangzhou, China.

Stanfelj, N., T.J. Marlin and J.F. Macgregor (1993). Monitoring nad Diagnosing Process Control Performance: The single-loop case. *Ind. Eng. Chem. Research* **32**(2), 301–314.

Tsiligiannis, C.A. and S.A. Svoronos (1988). Dynamic Interactors in Multivariable Process Control-II. Time Delays and Zeros Outside the Unit Circle. *Chemical Engineering Science* **43**(2), 339–347.

Tyler, M.L. and M. Morari (1996). Performance Monitoring Using Likelihood Methods. *Automatica* **36**, 1145–1162.

Van Den Hof, P.M.J. and R.J.P. Schrama (1993). An Indirect Method for Transfer Function Estimation from Closed Loop Data. *Automatica* **29**(6), 1523–1527.

Venugopal, R. and D.S. Bernstein (1997*a*). Adaptive Disturbance Rejection Using ARMarkov/Toeplitz Models. In: *Proceedings of the American Control Conference*. San Diego, California, USA. pp. 1657–1661.

Venugopal, R. and D.S. Bernstein (1997*b*). Adaptive Disturbance Rejection Using ARMarkov System Representations. In: *Proceedings of the 36th Conference on Decision and Control*. San Diego, California, USA. pp. 1884–1889.

Venugopal, R and D.S. Bernstein (2000). Adaptive Disturbance Rejection Using ARMARKOV/Toeplitz Models. *IEEE-Transactions-on-Control-Systems-Technology* **8**(2), 257–269.

Wahlberg, B. and L. Ljung (1986). Design variables for bias distribution in transfer function estimation. *IEEE Transactions on Automatic Control* **AC-31**, 134–144.

Wolowich, W.A. and P.L. Falb (1976). Invariants and canonical forms under dynamic compensation. *SIAM J. Control Optimiz.* **14**, 996–1008.

Yedavalli, R.K. (1985). Perturbation bounds for robust stability in linear state space models. *International Journal of Control* **42**(6), 1507–1517.

Appendix A

Factorization of Matrices in the AUDI Formulation

A.1 LDL^T and UDU^T factorization for the DPMM and IAM

The data product moment matrix (DPMM) $S_n(k)$, formed using the data vector $\phi_{na}(k)$ is expressed as (Niu *et al.* 1990, Niu *et al.* 1992, Banerjee and Shah 1996)

$$S_n(k) = \left[\sum_{i=1}^{k} \phi_{na}(i,n)\,\phi_{na}^T(i,n)\right]_{(2n+1)\times(2n+1)} \tag{A.1}$$

$$= \begin{bmatrix} \sum_{i=1}^{k} \phi_n(i,n)\,\phi_n^T(i,n) & \sum_{i=1}^{k} \phi_{na}(i,n)\,y(i) \\ \sum_{i=1}^{k} \phi_n^T(i,n)\,y(i) & \sum_{i=1}^{k} y(i)^2 \end{bmatrix} \tag{A.2}$$

Writing

$$A = \sum_{i=1}^{k} \phi_n(i,n)\,\phi_n^T(i,n) = S_{n-1}(k)$$

$$B = \sum_{i=1}^{k} \phi_{na}(i,n)\,y(i)$$

$$D = \sum_{i=1}^{k} y(i)^2$$

$$S_n(k) = \begin{bmatrix} A & B \\ B^T & D \end{bmatrix}$$

and the LDL^T decomposition of $S_n(k)$ is

$$S_n(k) = \begin{bmatrix} I_{2n} & 0 \\ B^T A^{-1} & I_{2n} \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} I_{2n} & 0 \\ B^T A^{-1} & I_{2n} \end{bmatrix}^T$$

where

$$\begin{aligned}\Delta &= D - B^T A^{-1} B = J_n \\ B^T A^{-1} &= \left[\sum_{i=1}^{k} \phi_{na}(i,n)\, y(i)\right]^T \left\{\sum_{i=1}^{k} \phi_n(i,n)\, \phi_n^T(i,n)\right\}^{-1} \\ &= \hat{\theta}^T(k)\end{aligned}$$

A is decomposed sequentially to obtain

$$S_n(k) = \left\{L_n(k)\, \Xi_n(k)\, [L_n(k)]^T\right\}$$

where

$$\begin{aligned}L_n(k) &= \begin{bmatrix} I_{2n} & 0 \\ \hat{\theta}^T(k) & I_{2n} \end{bmatrix} \begin{bmatrix} I_{2n-1} & 0 & 0 \\ \widehat{\alpha}_0^T(k-1) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & 0 & 0 \\ \widehat{\theta}_0^T(k-n) & 1 & 0 \\ 0 & 0 & I_{2n-1} \end{bmatrix} \\ \Xi_n(k) &= diag\{J_0(k-n), L_0(k-n), \cdots, J_{n-1}(k-1), L_{n-1}(k-1), J_n(k)\}\end{aligned}$$

Now

$$\begin{aligned}C_n(k) &= S_n(k)^{-1} \\ &= \left\{L_n(k)\, \Xi_n(k)\, [L_n(k)]^T\right\}^{-1} \\ &= \left\{L_n^{-T}(k)\, \Xi_n(k)^{-1}\, L_n^{-1}(k)\right\} \\ &= U_n(k)\, D_n(k)\, U_n^T(k)\end{aligned}$$